

开发有趣的极客项目，掌握实用的Python编程技能

异步图书
www.epubit.com.cn



Python极客项目编程

PYTHON PLAYGROUND

[美] Mahesh Venkitachalam 著 王海鹏 译



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS



[□□□□](#)

[□□□□](#)

[□□□□](#)

[□□](#)

[□□](#)

[□□□□ □□□□](#)

[□1□ □□iTunes□□□□](#)

[1.1 iTunes□□□□□□□□](#)

[1.2 □□□□](#)

[1.3 □□](#)

[1.3.1 □□□□](#)

[1.3.2 □□□□](#)

[1.3.3 □□□□□□□□□□□□□□](#)

[1.3.4 □□□□□□](#)

[1.3.5 □□□□](#)

1.3.6 □□□□□

1.4 □□□□

1.5 □□□□

1.6 □□

1.7 □□

2□ □□□

2.1 □□□□

2.1.1 □□□□□

2.1.2 □□□□

2.2 □□□□

2.3 □□

2.3.1 Spiro□□□□

2.3.2 □□□□

2.3.3 restart()□□

2.3.4 draw()□□

2.3.5 □□□□

2.3.6 SpiroAnimator

2.3.7 genRandomParams()

2.3.8

2.3.9 update()

2.3.10

2.3.11

2.3.12

2.4

2.5

2.6

2.7

3 Conway

3.1

3.2

3.3

3.3.1 音源

3.3.2 音源

3.3.3 音源

3.3.4 音源

3.3.5 音源

3.3.6 音源

3.4 音源

3.5 音源

3.6 音源

3.7 音源

4 音源 Karplus-Strong 音源

4.1 音源

4.1.1 音源

4.1.2 音源 WAV 音源

4.1.3 音源

4.2 音源

4.3 □□

4.3.1 □deque□□□□□□□

4.3.2 □□Karplus-Strong□□

4.3.3 □WAV□□

4.3.4 □pygame□□WAV□□

4.3.5 main()□□

4.4 □□□□

4.5 □□□□□□□

4.6 □□

4.7 □□

□5□ □□□□□□□□□□

5.1 □□□□

5.2 □□□□

5.3 □□

5.3.1 □□□□□□□□□□□□

5.3.2 □□□□□□□

[5.3.3 00000](#)

[5.3.4 0000000](#)

[5.3.5 0000](#)

[5.3.6 00000](#)

[5.3.7 00000](#)

[5.3.8 Boids](#)

[5.4 0000](#)

[5.5 00000000](#)

[5.6 00](#)

[5.7 00](#)

[0000 0000](#)

[6 ASCII0000](#)

[6.1 0000](#)

[6.2 0000](#)

[6.3 00](#)

[6.3.1 0000000000](#)

6.3.2 文字列

6.3.3 文字列ASCII

6.3.4 文字列

6.3.5 ASCII文字列

6.4 文字列

6.5 ASCII文字列

6.6 文字列

6.7 文字列

7 文字列

7.1 文字列

7.1.1 文字列

7.1.2 文字列

7.1.3 文字列

7.2 文字列

7.3 文字列

7.3.1 文字列

7.3.2 □□□□□□□□□□□□

7.3.3 □□□□□□□□□□

7.3.4 □□□□□□□□□□

7.3.5 □□□□□□□

7.3.6 □□□□□□□□

7.3.7 □□□□□□□□

7.3.8 □□□□□□□□□□□□

7.4 □□□□□

7.5 □□□□□□□□□□□□

7.6 □□

7.7 □□

□8□ □□□□□□

8.1 □□□□□

8.1.1 □□□□□□□□□□□□

8.1.2 □□□□

8.2 □□□□□

8.3 関数

8.3.1 関数の宣言と定義

8.3.2 関数の呼び出し

8.3.3 関数の戻り値

8.3.4 関数のスコープ

8.4 変数

8.5 配列

8.6 文字列

8.7 構造体

9章 OpenGL

9.1 OpenGL

9.2 OpenGLの初期化

9.2.1 初期化

9.2.2 初期化

9.2.3 初期化

[9.2.4 矩形](#)

[9.2.5 圆](#)

[9.2.6 使用OpenGL](#)

[9.3 矩形](#)

[9.4 圆](#)

[9.4.1 使用OpenGL](#)

[9.4.2 矩形](#)

[9.4.3 Scene](#)

[9.5 矩形](#)

[9.6 使用OpenGL](#)

[9.7 圆](#)

[9.8 圆](#)

[10 矩形](#)

[10.1 矩形](#)

[10.1.1 矩形](#)

[10.1.2 矩形](#)

10.1.3 関数

10.1.4 OpenGLのインストール

10.1.5 初期設定

10.1.6 実行

10.2 関数

10.3 関数

10.3.1 関数

10.3.2 関数

10.3.3 関数

10.3.4 関数

10.3.5 関数

10.3.6 関数

10.3.7 Camera

10.4 関数

10.5 関数

10.6 関数

[10.6.1 矩形区域](#)

[10.6.2 矩形区域](#)

[10.6.3 矩形区域](#)

[10.7 矩形区域](#)

[10.8 矩形区域](#)

[10.9 矩形](#)

[10.10 矩形](#)

[11.1 矩形](#)

[11.1 矩形](#)

[11.1.1 矩形](#)

[11.1.2 矩形](#)

[11.1.3 矩形OpenGL](#)

[11.2 矩形](#)

[11.3 矩形](#)

[11.4 矩形](#)

[11.5 矩形](#)

11.6 関数

11.6.1 関数の定義

11.6.2 関数の呼び出し

11.6.3 関数の戻り値

11.6.4 関数の引数

11.6.5 関数のスコープ

11.6.6 関数の再帰

11.7 配列

11.8 文字列

11.8.1 文字列の宣言

11.8.2 文字列の初期化

11.9 構造体

11.10 unions

11.10.1 unionsの宣言

11.10.2 unionsの初期化

11.10.3 unionsの使い方

11.11 〇〇〇〇〇〇〇〇〇〇

11.12 〇〇〇〇

11.13 〇〇〇〇〇〇〇〇〇

11.14 〇〇〇〇

11.15 〇〇

11.16 〇〇

〇〇〇〇 〇〇〇〇

12 Arduino

12.1 Arduino

12.2 Arduino

12.2.1 〇〇

12.2.2 IDE

12.2.3 〇〇

12.2.4 〇〇

12.3 〇〇〇〇

12.4 〇〇〇〇〇〇

[12.4.1 〇〇〇〇〇〇](#)

[12.4.2 Arduino〇〇](#)

[12.4.3 〇〇〇〇〇〇](#)

[12.5 Python〇〇](#)

[12.6 〇〇〇Python〇〇](#)

[12.7 〇〇〇〇](#)

[12.8 〇〇](#)

[12.9 〇〇](#)

[〇13〇 〇〇〇〇〇〇](#)

[13.1 〇〇〇〇〇〇〇〇](#)

[13.1.1 〇〇〇〇](#)

[13.1.2 〇〇〇〇〇〇〇〇](#)

[13.2 〇〇〇〇](#)

[13.2.1 〇〇〇〇〇](#)

[13.2.2 〇〇〇〇〇〇〇〇](#)

[13.3 Arduino〇〇](#)

[13.3.1 Arduino](#)

[13.3.2](#)

[13.3.3](#)

[13.4 Python](#)

[13.4.1](#)

[13.4.2](#)

[13.4.3 FFT](#)

[13.4.4 FFT](#)

[13.4.5](#)

[13.4.6](#)

[13.4.7](#)

[13.4.8](#)

[13.5 Python](#)

[13.6](#)

[13.7](#)

[13.8](#)

14 環境構築

14.1 準備

14.1.1 DHT11湿度温度センサ

14.1.2 温度センサ

14.1.3 湿度センサ

14.2 環境構築

14.2.1 環境構築

14.2.2 環境構築

14.2.3 Wi-Fi環境

14.2.4 環境構築

14.2.5 環境SSH環境

14.2.6 Web環境Bottle

14.2.7 環境flot環境

14.2.8 環境環境

14.3 環境構築

14.4 環境

[14.4.1 〇〇〇〇〇〇〇〇〇〇〇](#)

[14.4.2 〇〇〇〇](#)

[14.4.3 update\(\)〇〇](#)

[14.4.4 〇〇LED〇JavaScript〇〇〇〇](#)

[14.4.5 〇〇〇〇〇〇](#)

[14.5 〇〇〇〇](#)

[14.6 〇〇〇〇](#)

[14.7 〇〇](#)

[14.8 〇〇](#)

[〇〇A 〇〇〇〇](#)

[〇〇B 〇〇〇〇〇〇〇〇](#)

[〇〇C 〇〇〇〇〇〇〇〇〇〇](#)

[〇〇〇〇〇〇〇〇〇〇](#)



〇〇〇Python〇〇〇〇〇〇

ISBN 978-7-115-44976-4

[illegible][illegible][illegible][illegible]

- [] Mahesh Venkitachalam

□ □ □ □

11111111111111111111

- [illegible]

100164 315@ptpress.com.cn

 <http://www.ptpress.com.cn>

- □□□□□□□(010)81055410

□□□□□□(010)81055315

Python 14 Python
Python
iTunes ASCII
Python Arduino
Python
Python Python
Python Python Python

Python
Python Python
Python

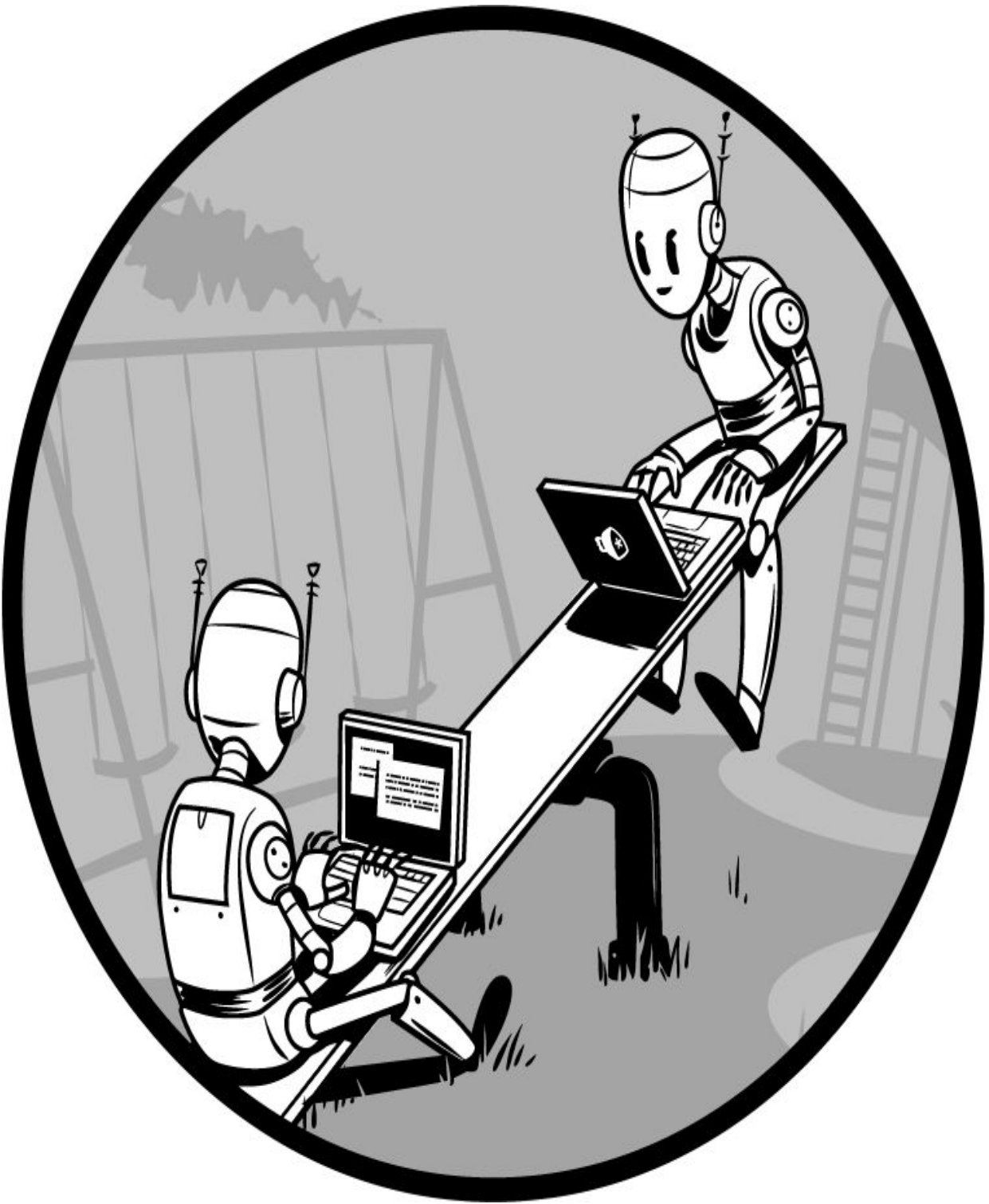


Hema
Raviprakash
Jayaraman “”
S.P. Road
Zoo Seby Kallarakkal
Santosh
Hemachandra
Karthikeyan Chellappa Python

□□□□□□□□Kaikondrahalli□□□□□□□□□Matthew
Denham□□□□□Reddit□□□□□□□□□□
□Spirograph□□□□□□□□□□□□□

□□□No Starch□□□□Tyler Ortman□Bill
Pollock□□□□□□□□□□□□□□□□□Serena Yang□
□□□□□□□□□□□□Nicholas Kramer□□□□□□□□□□
□□□□□□□A.V. Venkitachalam□N. Saraswathy□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□





□□□□□□□□□□□□□□□□14□□□□□□□□□□□□□□□□
Python□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

ASCII 3D
Python

Python

Python

Python
Python
Python
Python
Python
Python
Python

Python

Python

Python

1 iTunes
2
Python

--	--	--	--	--	--	--	--	--

3 Conway
 4 Karplus-Strong
 5

Python 2D 6
ASCII 7 8
3D

--	--	--	--	--	--	--	--	--

☐ OpenGL 3D ☐ 9

☐ OpenGL 3D ☐ 10

☐ OpenGL ☐ 11

☐ OpenGL

☐ MRI CT

--	--	--	--	--	--	--	--	--

Python Arduino
12 Arduino
13 Python Arduino

14

Python

Python

Python

```

C C ++
Python
Python Python
Python Python
Python

```

`Python`

Python
for while
Python
Python
Python
Python
C/C++ Python

Python
Python
Python
Arduino JavaScript
Python

Python 14
/

00110111000000000000110100000000001010001

Python

```
>>> str = '00110111000000000000110100000000001010001'
```

```
>>> len(str)
```

40

```
>>> [int(str[i:i+8], 2) for i in range(0, 40, 8)]
```

[55, 0, 26, 0, 81]

40 5 8
55.0% 26.0 55 + 26
= 81

Python
Python

Python

Python 3.3.3Python 2.7.x
3.x

<https://github.com/electronut/pp/>

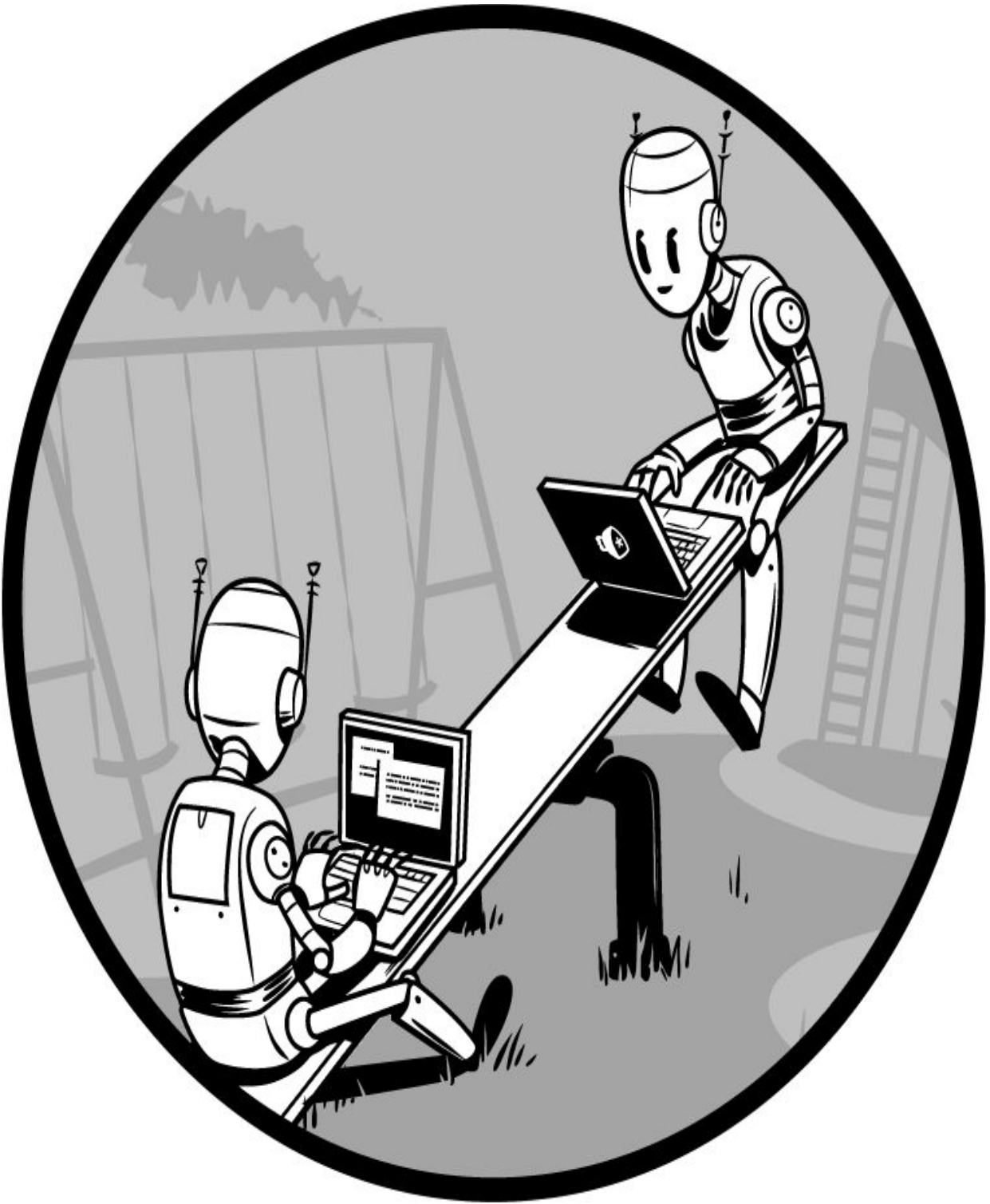
www.epubit.com.cn

Download Zip

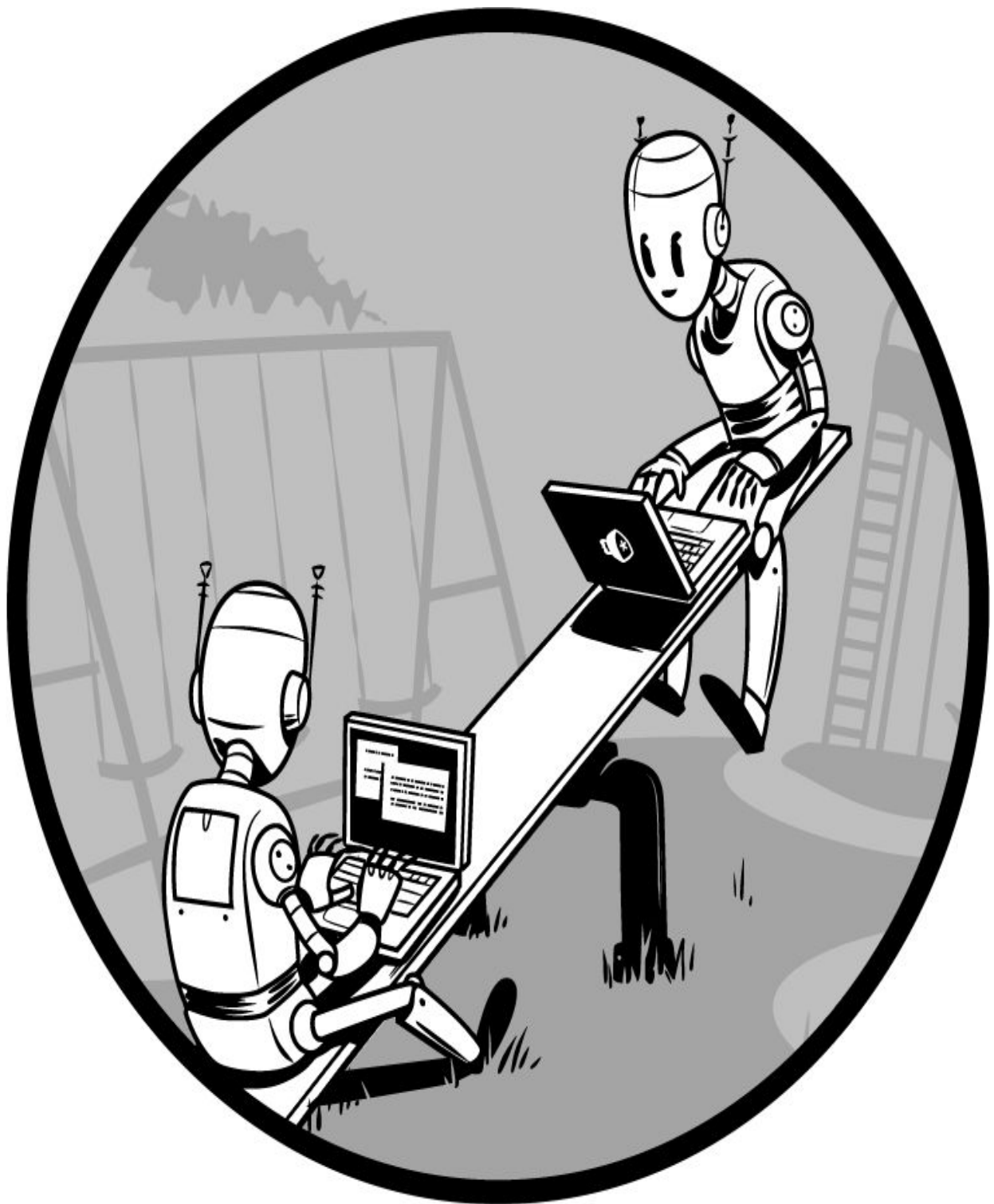
“

”

——



1 iTunes



Python iTunes
iTunes Python
matplotlib

- XML p-list
- Python
- Python set
- numpy
-
- matplotlib
-

1.1 iTunes

iTunes iTunes
File►Library►Export Playlist
XML
<MyTag>

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
❶ <!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0/  
/EN" "http://www
```

```
apple.com/DTDs/PropertyList-1.0.dtd">
```


❷ <plist version="1.0">

❸ <dict>

❹ <key>Major Version</key><integer>1</integer>

<key>Minor Version</key><integer>1</integer>

-- *snip* --

❺ <key>Tracks</key>

<dict>

<key>2438</key>

<dict>

<key>Track ID</key><integer>2438</integer>

<key>Name</key><string>Yesterday</string>

<key>Artist</key><string>The Beatles</string>

<key>Composer</key>

<string>Lennon [John], McCartney [Paul]</string>

<key>Album</key><string>Help!</string>

</dict>

-- *snip* --

</dict>

❻ <key>Playlists</key>

<array>

<dict>

<key>Name</key><string>Now</string>

<key>Playlist ID</key>

```

<integer>21348</integer>

    -- snip --

    <array>

        <dict>

            <key>Track ID</key>
            <integer>6382</integer>

        </dict>

    -- snip --

    </array>

</dict>

</array>

</dict>

</plist>

```

plist P-list plist dict key

<xml> XML DTD XML ① URL

② <plist> <dict> ③ ④ Major Version Minor Version ⑤ Tracks

1.4

1.3.1

`findDuplicates()`

```
def findDuplicates(fileName):  
    print('Finding duplicate tracks in %s...' % fileName)  
    # read in a playlist  
    ❶ plist = plistlib.readPlist(fileName)  
    # get the tracks from the Tracks dictionary  
    ❷ tracks = plist['Tracks']  
    # create a track name dictionary  
    ❸ trackNames = {}  
    # iterate through the tracks  
    ❹ for trackId, track in tracks.items():  
        try:  
            ❺ name = track['Name']  
            duration = track['Total Time']  
            # look for existing entries  
            ❻ if name in trackNames:  
                # if a name and duration match, increment  
                the count  
                # round the track length to the nearest se  
                cond
```

```

    ⑦ if duration//1000 == trackNames[name]
    [0]//1000:

        count = trackNames[name][1]

    ⑧ trackNames[name] = (duration, count+1
    )

    else:

        # add dictionary entry as tuple (duration,
    count)

    ⑨ trackNames[name] = (duration, 1)

    except:

        # ignore

        pass

```

❶ readPlist() returns a plist object
 ❷ Tracks is a dictionary
 ❸ items() returns a list of items
 ❹ Python

❺ in
 ❻
 ❼ // 1000
 name
 duration count ⑧
 count 1 ⑨

```
for i in trackNames:
    try:
        # do something
    except:
        pass
```

1.3.2 Duplicates

Find duplicate track names

```
# store duplicates as (name, count) tuples

❶ dups = []

for k, v in trackNames.items():

❷     if v[1] > 1:

        dups.append((v[1], k))

# save duplicates to a file

❸ if len(dups) > 0:

        print("Found %d duplicates. Track names saved to dups.txt" % len(dups))

    else:

        print("No duplicate tracks found!")

❹ f = open("dups.txt", "w")

for val in dups:

❺     f.write("[%d] %s\n" % (val[0], val[1]))

f.close()
```

❶ Iterate over trackNames items to find duplicates
trackNames.items() returns a list of (trackName, count) tuples

1 2 name count
3 open() 4
5 dups

1.3.3

```
def findCommonTracks(fileNames):  
    # a list of sets of track names  
    ❶ trackNameSets = []  
    for fileName in fileNames:  
        # create a new set  
        ❷ trackNames = set()  
        # read in playlist  
        ❸ plist = plistlib.readPlist(fileName)  
        # get the tracks  
        tracks = plist['Tracks']  
        # iterate through the tracks  
        for trackId, track in tracks.items():  
            try:  
                # add the track name to a set  
                ❹ trackNames.add(track['Name'])  
            except:  
                # ignore
```

```

        pass

    # add to list

    ⑤ trackNameSets.append(trackNames)

    # get the set of common tracks

    ⑥ commonTracks = set.intersection(*trackNameSets)

    # write to file

    if len(commonTracks) > 0:

    ⑦     f = open("common.txt", "w")

        for val in commonTracks:

            s = "%s\n" % val

    ⑧     f.write(s.encode("UTF-8"))

        f.close()

        print("%d common tracks found. "

              "Track names written to common.txt." % len(c
ommonTracks))

    else:

        print("No common tracks!")

```

findCommonTracks()
 ①
 trackNames Python
 set ② findDuplicates() plistlib
 ③ Tracks

trackNames ④
trackNameSets ⑤

⑥ set.intersection()
Python*
⑦
encode() Unicode
⑧

1.3.4

plotStats()

```
def plotStats(fileName):  
    # read in a playlist  
    ❶ plist = plistlib.readPlist(fileName)  
    # get the tracks from the playlist  
    tracks = plist['Tracks']  
    # create lists of song ratings and track durations  
    ❷ ratings = []  
    durations = []  
    # iterate through the tracks  
    for trackId, track in tracks.items():  
        try:  
            ❸ ratings.append(track['Album Rating'])  
            durations.append(track['Total Time'])
```


- ⑤ `pyplot.plot(x, y, 'o')`
- ⑥ `pyplot.axis([0, 1.05*np.max(x), -1, 110])`
- ⑦ `pyplot.xlabel('Track duration')`
- ⑧ `pyplot.ylabel('Track rating')`

`# plot histogram`

`pyplot.subplot(2, 1, 2)`

- ⑨ `pyplot.hist(x, bins=20)`
- `pyplot.xlabel('Track duration')`
- `pyplot.ylabel('Count')`

`# show plot`

- ⑩ `pyplot.show()`

① `numpy.array()` `np`
`32` ② `numpy`
`60×1000` ③ `numpy`
`y`

`matplotlib` ④ `subplot()` `(2, 1, 1)` `matplotlib`
`2` `1` `1` ⑤ `plot()` `o` `matplotlib`

⑥ `x` `y` 列表的列表形式
⑦ ⑧ `x` `y` 列表的列表形式

`matplotlib` `hist()` 函数
⑨ `bins` 参数
`show()` ⑩ `matplotlib`

1.3.6 主函数

`main()`

```
def main():  
    # create parser  
    descStr = """  
        This program analyzes playlist files (.xml) exported from iTunes.  
        """  
    ❶ parser = argparse.ArgumentParser(description=descStr)  
    # add a mutually exclusive group of arguments  
    ❷ group = parser.add_mutually_exclusive_group()  
  
    # add expected arguments  
    ❸ group.add_argument('--common', nargs='*', dest='plFiles', required=False)  
    ❹ group.add_argument('--stats', dest='plFile', required=False)
```

```
⑤ group.add_argument('--  
dup', dest='plFileD', required=False)
```

```
# parse args
```

```
⑥ args = parser.parse_args()
```

```
if args.plFiles:
```

```
# find common tracks
```

```
findCommonTracks(args.plFiles)
```

```
elif args.plFile:
```

```
# plot stats
```

```
plotStats(args.plFile)
```

```
elif args.plFileD:
```

```
# find duplicate tracks
```

```
findDuplicates(args.plFileD)
```

```
else:
```

```
⑦ print("These are not the tracks you are looking f  
or.")
```

```
Python argparse ①  
ArgumentParser  
argparse
```

parser.add_argument(2)

parser.add_mutually_exclusive_group()
parser.add_argument()

3 4 5

args.plFiles args.plFile args.plFileD

6

findCommonTracks() plotStats()

findDuplicates()

args

--common

args.plFiles None

7

1.4

<https://github.com/electronut/pp/tree/master/playlist/>

```
import re, argparse
```

```
import sys
```

```
from matplotlib import pyplot
```

```
import plistlib
```

```
import numpy as np
```

```

def findCommonTracks(fileNames):
    """
    Find common tracks in given playlist files,
    and save them to common.txt.
    """

    # a list of sets of track names
    trackNameSets = []

    for fileName in fileNames:
        # create a new set
        trackNames = set()

        # read in playlist
        plist = plistlib.readPlist(fileName)

        # get the tracks
        tracks = plist['Tracks']

        # iterate through the tracks
        for trackId, track in tracks.items():
            try:
                # add the track name to a set
                trackNames.add(track['Name'])
            except:
                # ignore
                pass

```

```

# add to list
trackNameSets.append(trackNames)

# get the set of common tracks
commonTracks = set.intersection(*trackNameSets)

# write to file
if len(commonTracks) > 0:
    f = open("common.txt", 'w')
    for val in commonTracks:
        s = "%s\n" % val
        f.write(s.encode("UTF-8"))
    f.close()
    print("%d common tracks found. "
          "Track names written to common.txt." % le
n(commonTracks))
else:
    print("No common tracks!")

def plotStats(fileName):
    """
    Plot some statistics by reading track information f
rom playlist.
    """
    # read in a playlist
    plist = plistlib.readPlist(fileName)

```



```

# get the tracks from the playlist
tracks = plist['Tracks']

# create lists of song ratings and track durations
ratings = []
durations = []

# iterate through the tracks
for trackId, track in tracks.items():
    try:
        ratings.append(track['Album Rating'])
        durations.append(track['Total Time'])
    except:
        # ignore
        pass

# ensure that valid data was collected
if ratings == [] or durations == []:
    print("No valid Album Rating/Total Time data in
%s." % fileName)
    return

# scatter plot
x= np.array(durations, np.int32)

# convert to minutes
x = x/60000.0

```

```

y = np.array(ratings, np.int32)
pyplot.subplot(2, 1, 1)
pyplot.plot(x, y, 'o')
pyplot.axis([0, 1.05*np.max(x), -1, 110])
pyplot.xlabel('Track duration')
pyplot.ylabel('Track rating')

```

```

# plot histogram
pyplot.subplot(2, 1, 2)
pyplot.hist(x, bins=20)
pyplot.xlabel('Track duration')
pyplot.ylabel('Count')
# show plot
pyplot.show()

```

```

def findDuplicates(fileName):

```

```

    """

```

```

    Find duplicate tracks in given playlist.

```

```

    """

```

```

    print('Finding duplicate tracks in %s...' % fileName)

```

e)

```

    # read in playlist

```

```

    plist = plistlib.readPlist(fileName)

```

```

# get the tracks from the Tracks dictionary
tracks = plist['Tracks']

# create a track name dictionary
trackNames = {}

# iterate through tracks
for trackId, track in tracks.items():
    try:
        name = track['Name']
        duration = track['Total Time']
        # look for existing entries
        if name in trackNames:
            # if a name and duration match, increme
nt the count
            # round the track length to the nearest
second
            if duration//1000 == trackNames[name]
[0]//1000:
                count = trackNames[name][1]
                trackNames[name] = (duration, count
+1)
            else:
                # add dictionary entry as tuple (durati
on, count)
                trackNames[name] = (duration, 1)
        except:

```

```

        # ignore
        pass

    # store duplicates as (name, count) tuples
    dups = []

    for k, v in trackNames.items():
        if v[1] > 1:
            dups.append((v[1], k))

    # save duplicates to a file

    if len(dups) > 0:
        print("Found %d duplicates. Track names saved to
dup.txt" % len(dups))
    else:
        print("No duplicate tracks found!")

    f = open("dups.txt", 'w')

    for val in dups:
        f.write("[%d] %s\n" % (val[0], val[1]))

    f.close()


# gather our code in a main() function
def main():

    # create parser

    descStr = """

    This program analyzes playlist files (.xml) exported

```

d from iTunes.

```
    """

    parser = argparse.ArgumentParser(description=descSt
r)

    # add a mutually exclusive group of arguments
    group = parser.add_mutually_exclusive_group()

    # add expected arguments

    group.add_argument('--
common', nargs='*', dest='plFiles', required=False)

    group.add_argument('--
stats', dest='plFile', required=False)

    group.add_argument('--
dup', dest='plFileD', required=False)


    # parse args

    args = parser.parse_args()


    if args.plFiles:

        # find common tracks

        findCommonTracks(args.plFiles)

    elif args.plFile:

        # plot stats

        plotStats(args.plFile)

    elif args.plFileD:
```

```

        # find duplicate tracks

        findDuplicates(args.plFileD)

    else:

        print("These are not the tracks you are looking for
        .")

# main method

if __name__ == '__main__':

    main()

```

1.5



```
$ python playlist.py --common test-data/maya.xml test-
data/rating.xml
```



5 common tracks found. Track names written to common.txt.

```
$ cat common.txt
```

God Shuffled His Feet

Rubric

Floe

Stairway To Heaven

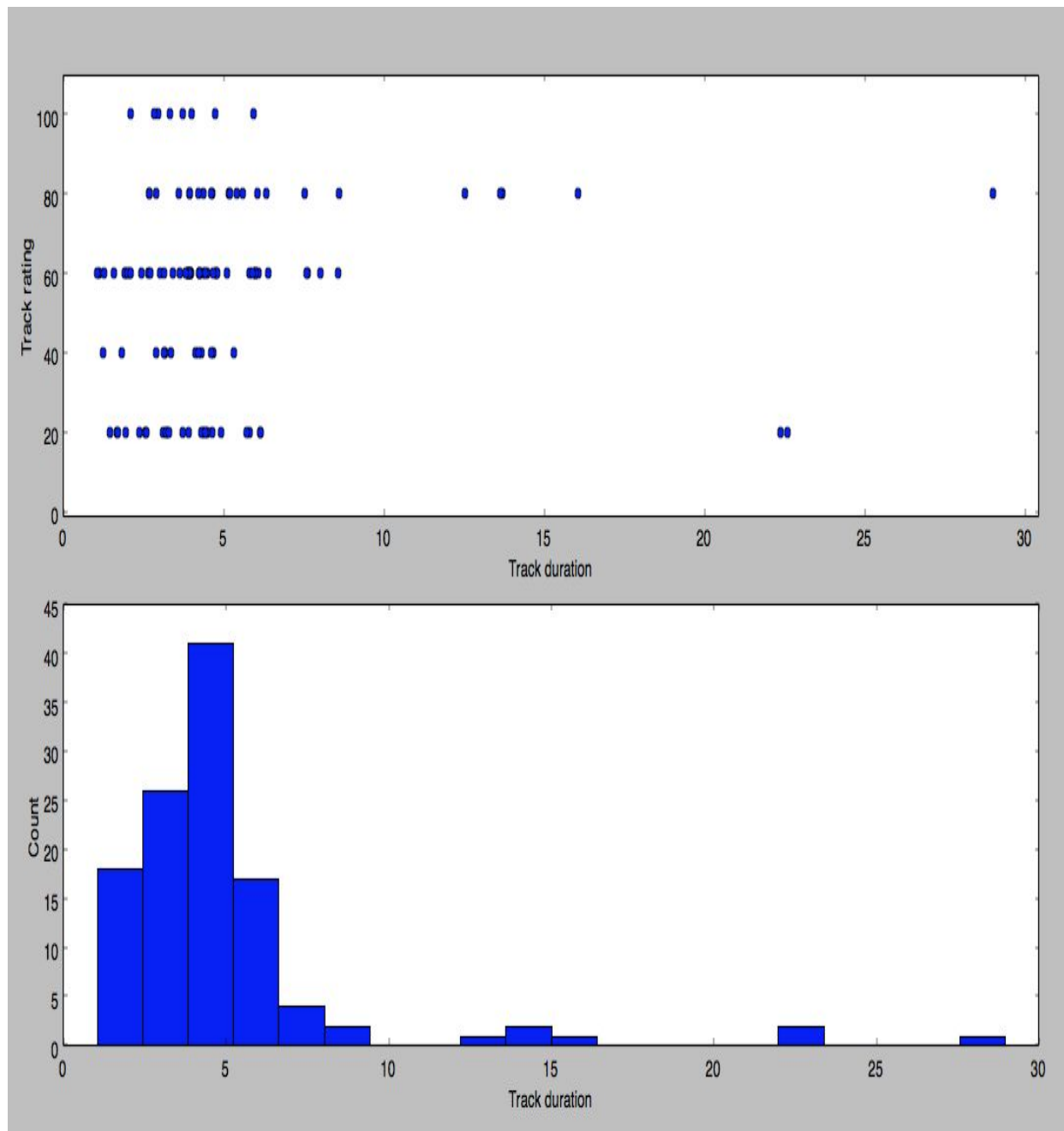
Pi's Lullaby

```
moksha:playlist mahesh$
```

□□□□□□□□□□□□□□□□□□□□

```
$ python playlist.py --stats test-data/rating.xml
```

□1-1□□□□□□□□□□□□



□1-1 playlist.py□□□□

1.6 数据

我们使用 iTunes 的 Python 接口来访问 iTunes 的数据库。Python 的 iTunes 接口是一个名为 `iTunes` 的模块，它提供了访问 iTunes 数据库的接口。Python 的 iTunes 接口是一个名为 `iTunes` 的模块，它提供了访问 iTunes 数据库的接口。

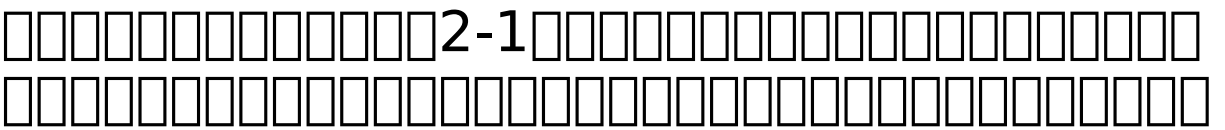
1.7 分析

我们将使用以下代码来访问 iTunes 数据库。

```
1 # 访问 iTunes 数据库
2 # 使用 findCommonTracks() 函数来查找共同的音乐
3 # findCommonTracks() 函数返回一个列表，其中包含共同的音乐名称
```

```
2 # plotStats() 函数使用 matplotlib 的 hist() 函数来绘制
3 # 共同音乐的直方图。hist() 函数返回一个列表，其中包含
4 # 共同音乐的直方图。matplotlib 的 hist() 函数返回一个列表，其中包含
```

```
3 # 共同音乐的直方图。matplotlib 的 hist() 函数返回一个列表，其中包含
4 # 共同音乐的直方图。matplotlib 的 hist() 函数返回一个列表，其中包含
5 # 共同音乐的直方图。matplotlib 的 hist() 函数返回一个列表，其中包含
```

Python 3.6.1
Windows 10

Python 3.6.1
spiro.py Python 3.6.1
PNG

Python 3.6.1

- turtle
-
-
-
-
-
-



□2-1 □□□

```

turtle
turtle

```

2.1

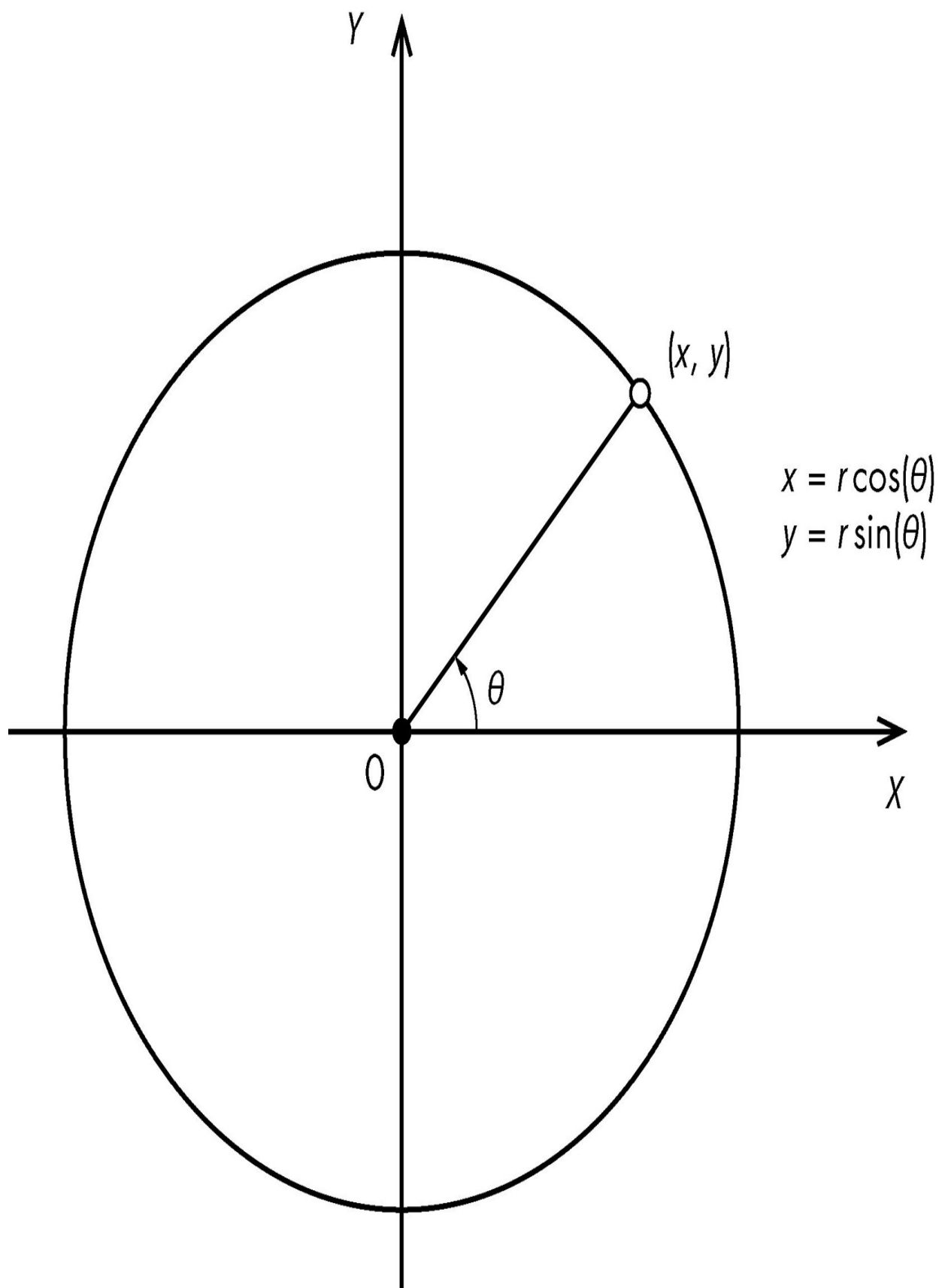
[illegible]
$$r \begin{pmatrix} x \\ y \end{pmatrix}$$
[illegible]

$$x = r \cos(\theta)$$

$$y = r \sin(\theta)$$

θ X Y
 $X^2 + Y^2 = R^2$ $\theta = 0$

$$2\pi \int_0^{2\pi} \int_0^1 \sqrt{x^2 + y^2 + 2} \, dy \, dx$$

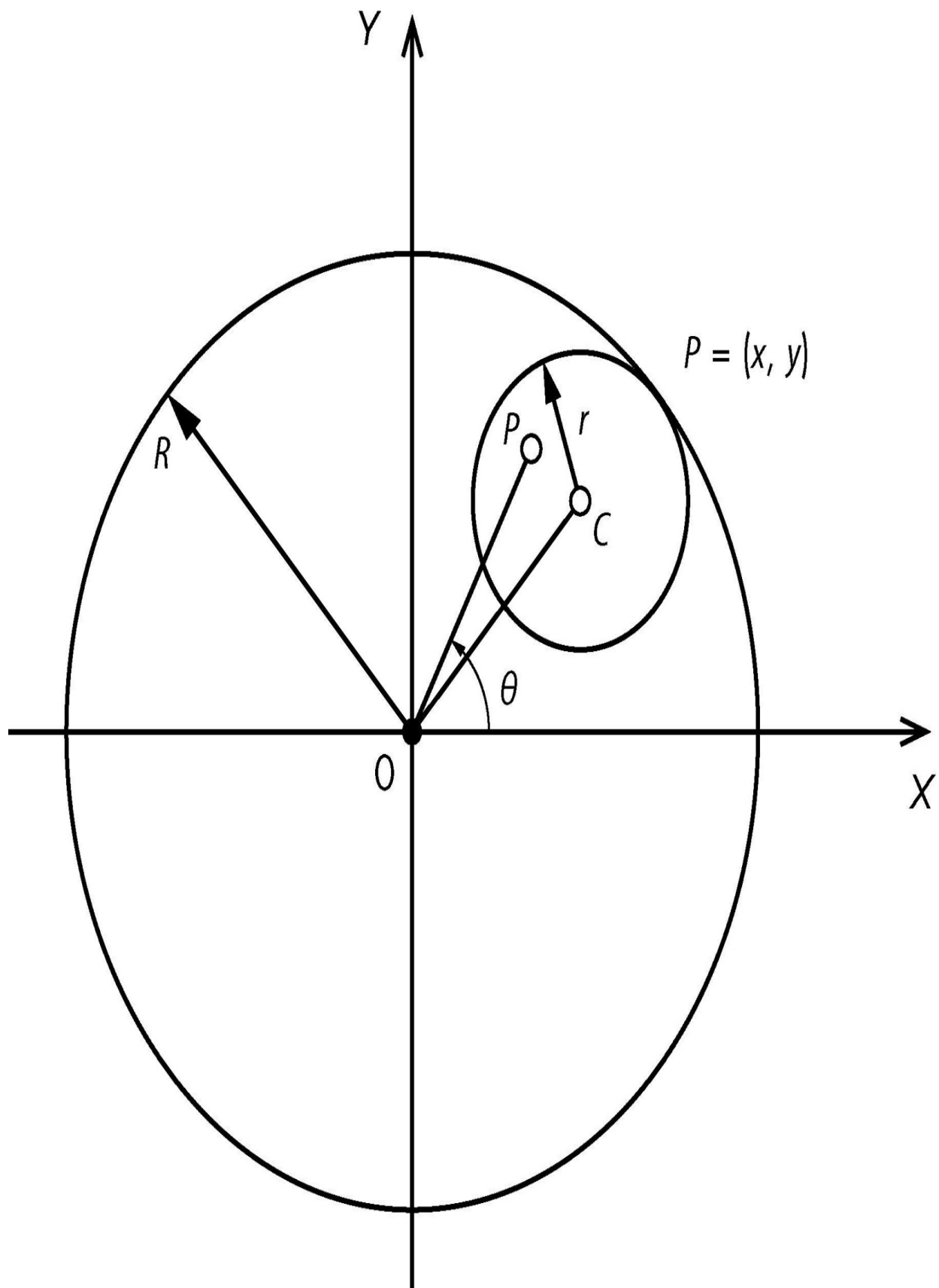


2-2 極座標方程式

平面上の任意の点 (a, b) を
極座標 (r, θ) で表すとき、 $x = a + r \cos(\theta)$ 、 $y = b + r \sin(\theta)$ の関係が成り立つ。
これを極座標方程式と呼ぶ。

2.1.1 極座標

2-3 極座標と直角座標の相互変換
極座標 (r, θ) と直角座標 (x, y) の関係は、
 $x = r \cos(\theta)$ 、 $y = r \sin(\theta)$ 、 $r = \sqrt{x^2 + y^2}$ 、 $\theta = \arctan\left(\frac{y}{x}\right)$ によって与えられる。



2-3 位置ベクトル

2-3 図 C は、位置ベクトル \mathbf{P} の終点の軌跡が半径 R の円、
中心 r の円周上を動く。

$$k = \frac{r}{R}$$

図 C は、位置ベクトル r の終点の軌跡が $|\mathbf{P}| = PC / r$ の円周上を
動く。位置ベクトル \mathbf{P} の終点の軌跡が半径 R の円、

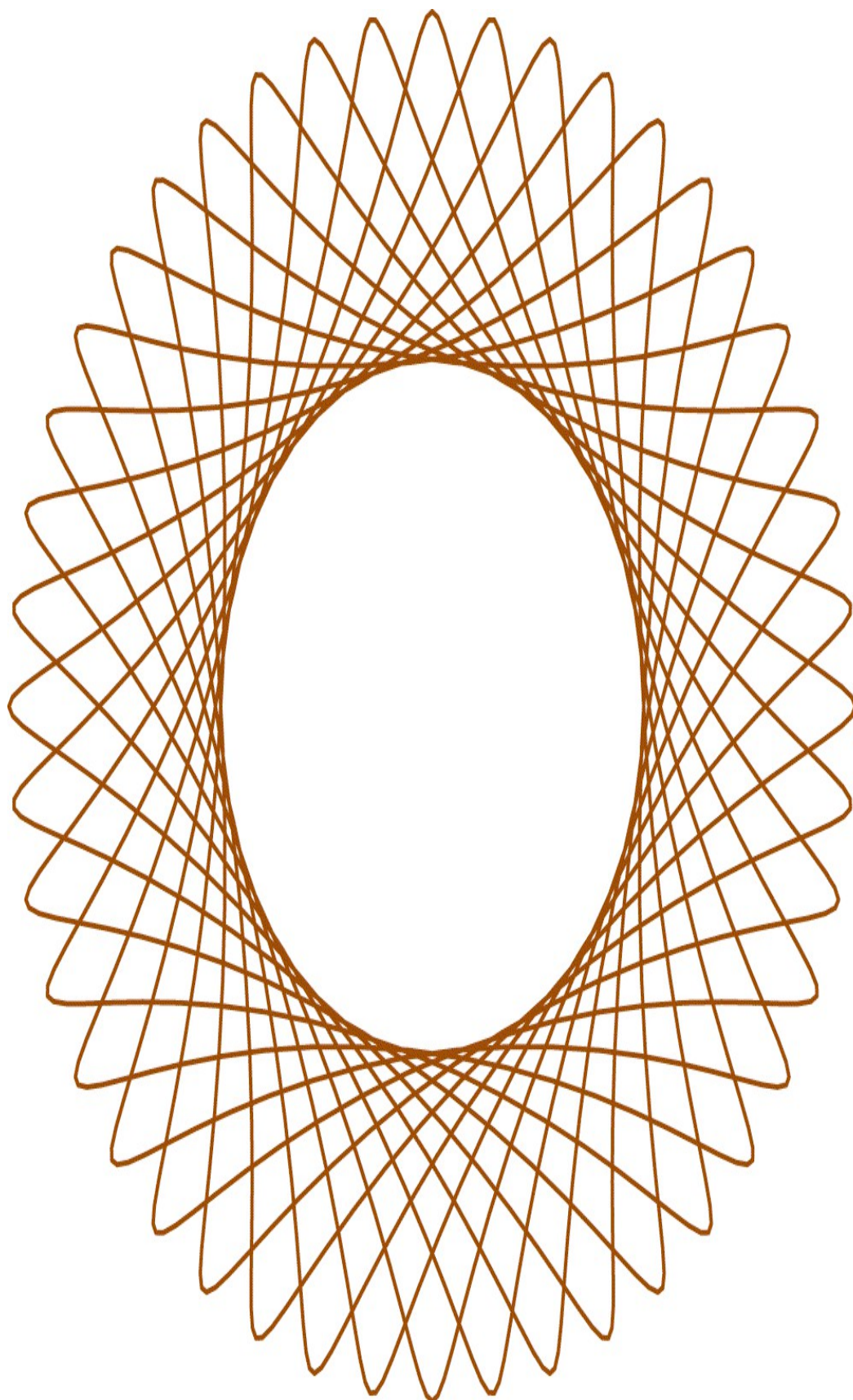
$$x = R \left((1 - k) \cos(\theta) + k \cos\left(\frac{1 - k}{k} \theta\right) \right)$$

$$y = R \left((1 - k) \sin(\theta) + k \sin\left(\frac{1 - k}{k} \theta\right) \right)$$

図

位置ベクトル \mathbf{P} の終点の軌跡が半径 R の円、
中心 r の円周上を動く [\[1\]](#)

2-4 図 C は、位置ベクトル \mathbf{P} の終点の軌跡が半径 R の円、
中心 r の円周上を動く。



$R = 220$

$r = 65$

$l = 0.8$

2-4 例题

已知一个半径为 R 的圆，其内接正 n 边形的边长为 a_n ，求该正 n 边形的面积 S_n 。

$$\frac{r}{R}$$

已知一个半径为 R 的圆，其内接正 n 边形的边长为 a_n ，求该正 n 边形的面积 S_n 。

$$\frac{r}{R} = \frac{65}{220}$$

已知一个半径为 R 的圆，其内接正 n 边形的边长为 a_n ，求该正 n 边形的面积 S_n 。

$$\frac{(65/5)}{(220/5)} = \frac{13}{44}$$

已知一个半径为 R 的圆，其内接正 n 边形的边长为 a_n ，求该正 n 边形的面积 S_n 。

已知一个半径为 R 的圆，其内接正 n 边形的边长为 a_n ，求该正 n 边形的面积 S_n 。

2.1.2 例题

Python turtle module
The turtle module is a standard Python module that allows you to draw various shapes and patterns using a turtle-like cursor. It is a simple and easy-to-use module that can be used to create a wide variety of graphics. The turtle module is a standard Python module that allows you to draw various shapes and patterns using a turtle-like cursor. It is a simple and easy-to-use module that can be used to create a wide variety of graphics.

drawcircle.py Python

```
import math

❶ import turtle

# draw the circle using turtle

def drawCircleTurtle(x, y, r):
    # move to the start of circle
    ❷ turtle.up()
    ❸ turtle.setpos(x + r, y)
    ❹ turtle.down()

    # draw the circle
    ❺ for i in range(0, 365, 5):
        ❻ a = math.radians(i)
        ❼ turtle.setpos(x + r*math.cos(a), y + r*math.sin(a))

    ❽ drawCircleTurtle(100, 100, 50)
```

⑨ turtle.mainloop()

① turtle 模块的 drawCircleTurtle() 方法
② turtle.up() 方法
Python 的 turtle 模块提供了很多方法，这里只介绍了几个。
其他方法请参考官方文档。

③ turtle.goto(x + r, y) 方法
④ turtle.down() 方法
⑤ range(0, 365, 5) 方法
5 表示每隔 5 度，i 从 0 到 360
⑥ turtle.circle(r) 方法
r 表示半径。

⑦ turtle.done() 方法
N 表示次数
N 表示次数

⑧ drawCircleTurtle() 方法
⑨ mainloop() 方法
tkinter 模块
Tkinter 是 Python 的 GUI 库

其他方法请参考官方文档。

2.2 总结

本章主要介绍了 turtle 模块。

- turtle 模块

- pillow is a Python module that wraps the PIL module

2.3 Turtle

The `Spiro` class is a subclass of `Turtle`. It has a `draw()` method that calls `update()` repeatedly to draw a Spirograph. The `Spiro` class is defined in the `SpiroAnimator` module.

The `Spiro` class is defined in the `2.4` module.

2.3.1 Spiro

The `Spiro` class is defined as follows:

```
# a class that draws a Spirograph
class Spiro:
    # constructor
    def __init__(self, xc, yc, col, R, r, l):

        # create the turtle object
        ❶ self.t = turtle.Turtle()

        # set the cursor shape
        ❷ self.t.shape('turtle')

        # set the step in degrees
        ❸ self.step = 5
```

```

        # set the drawing complete flag
④        self.drawingComplete = False

        # set the parameters
⑤        self.setparams(xc, yc, col, R, r, l)

        # initialize the drawing
⑥        self.restart()

```

❶ Spiro turtle
 ❷
 <https://docs.python.org/3.3/library/turtle.html>
 ❸
 ❹
 ❺
 ❻

❺ ❻

2.3.2

getParams() Spiro

```

        # set the parameters

        def setparams(self, xc, yc, col, R, r, l):

            # the Spirograph parameters
    ❶        self.xc = xc

```

```

        self.yc = yc
❷        self.R = int(R)

        self.r = int(r)

        self.l = l

        self.col = col

        # reduce r/R to its smallest form by dividing with
the GCD
❸        gcdVal = gcd(self.r, self.R)
❹        self.nRot = self.r//gcdVal

        # get ratio of radii

        self.k = r/float(R)

        # set the color

        self.t.color(*col)

        # store the current angle

❺        self.a = 0

```

❶ 初始化半径和中心点
 ❷ 将半径 R 和 r 转换为整数
 ❸ 使用 Python 的 fractions 模块中的 gcd() 函数计算 GCD
 ❹ 计算 self.nRot
 ❺ 初始化角度 a

2.3.3 restart()

restart() 方法用于重新启动 Spiro 对象的动画。


```

# restart the drawing

def restart(self):
    # set the flag
❶ self.drawingComplete = False

    # show the turtle
❷ self.t.showturtle()

    # go to the first point
❸ self.t.up()

❹ R, k, l = self.R, self.k, self.l

    a = 0.0

❺ x = R*((1-k)*math.cos(a) + l*k*math.cos((1-
k)*a/k))

    y = R*((1-k)*math.sin(a) - l*k*math.sin((1-
k)*a/k))

❻ self.t.setpos(self.xc + x, self.yc + y)

❼ self.t.down()

```

drawingComplete
 ❶ Spiro
 ❷ ❸
 ❹ ❺ a 0 x y
 ❻ ❼
 Setpos()

2.3.4 draw()

draw()

```
# draw the whole thing

def draw(self):

    # draw the rest of the points

    R, k, l = self.R, self.k, self.l

    ❶ for i in range(0, 360*self.nRot + 1, self.step):

        a = math.radians(i)

        ❷ x = R*((1-k)*math.cos(a) + l*k*math.cos((1-
k)*a/k))

        y = R*((1-k)*math.sin(a) - l*k*math.sin((1-
k)*a/k))

        self.t.setpos(self.xc + x, self.yc + y)

        # drawing is now done so hide the turtle cursor

        ❸ self.t.hideturtle()
```

❶ i from 0 to $360 \times \text{nRot}$

❷ X Y

❸ hide the cursor

2.3.5

update()

```
# update by one step

def update(self):

    # skip the rest of the steps if done
```

```

❶      if self.drawingComplete:
            return

            # increment the angle

❷      self.a += self.step

            # draw a step

            R, k, l = self.R, self.k, self.l

            # set the angle

❸      a = math.radians(self.a)

            x= self.R*((1-k)*math.cos(a) + l*k*math.cos((1-
k)*a/k))

            y = self.R*((1-k)*math.sin(a) - l*k*math.sin((1-
k)*a/k))

            self.t.setpos(self.xc + x, self.yc + y)

            # if drawing is complete, set the flag

❹      if self.a >= 360*self.nRot:

                self.drawingComplete = True

                # drawing is now done so hide the turtle curso
r

                self.t.hideturtle()

```

❶ update() drawingComplete
 ❷ update()
 ❸ X Y
 r

#####drawingComplete#####
#####

2.3.6 SpiroAnimator

SpiroAnimator#####

#####

```
# a class for animating Spirographs

class SpiroAnimator:

    # constructor

    def __init__(self, N):

        # set the timer value in milliseconds

        ❶ self.deltaT = 10

        # get the window dimensions

        ❷ self.width = turtle.window_width()

        self.height = turtle.window_height()

        # create the Spiro objects

        ❸ self.spiros = []

        for i in range(N):

            # generate random parameters

            ❹ rparams = self.genRandomParams()
```

```

        # set the spiro parameters

    ⑤        spiro = Spiro(*rparams)

            self.spiros.append(spiro)

        # call timer

    ⑥        turtle.ontimer(self.update, self.deltaT)

```

① SpiroAnimator 的 DeltaT 为 10
 ② 在 SpiroAnimator 中生成随机参数
 ③ 在 SpiroAnimator 中生成 Spiro 对象
 ④ 在 SpiroAnimator 中生成 Spiro 对象
 ⑤ 在 Spiro 对象中生成 Spiro 对象
 rparams 是 Spiro 对象的参数
 Python 的 * 运算符

⑥ turtle.ontimer() 的 DeltaT 为
 update()

④ genRandomParams() 生成随机参数

2.3.7 genRandomParams()

genRandomParams() 生成随机参数
 Spiro 对象

```

# generate random parameters

def genRandomParams(self):
    width, height = self.width, self.height

```

```

❶ R = random.randint(50, min(width, height)//2)
❷ r = random.randint(10, 9*R//10)
❸ l = random.uniform(0.1, 0.9)
❹ xc = random.randint(-width//2, width//2)
❺ yc = random.randint(-height//2, height//2)
❻ col = (random.random(),
          random.random(),
          random.random())
❼ return (xc, yc, col, R, r, l)

```

Python random randint() uniform()
 ❶ R 50
 ❷ r R 10% 90%

❸ l 0.1 0.9
 ❹ ❺ x y
 ❻
 ❼

2.3.8

restart()

```

# restart spiro drawing

def restart(self):
    for spiro in self.spiros:

```

```

# clear
spiro.clear()

# generate random parameters
rparams = self.genRandomParams()

# set the spiro parameters
spiro.setparams(*rparams)

# restart drawing
spiro.restart()

```

Spiro

2.3.9 update()

SproAnimator
 update()
 Spiro

```

def update(self):
    # update all spiros
    ❶ nComplete = 0
    for spiro in self.spiros:
        # update
        ❷ spiro.update()
        # count completed spiros
        ❸ if spiro.drawingComplete:
            nComplete += 1

```

```

        # restart if all spiros are complete
    ④    if nComplete == len(self.spiros):
            self.restart()

        # call the timer
    ⑤    turtle.ontimer(self.update, self.deltaT)

```

update() 检查 nComplete 是否等于 Spiro 对象的个数 ①。如果等于，则调用 restart() 方法 ②。然后，调用 Spiro 对象的 update() 方法 ③，并设置 timer 为 1。

最后，调用 restart() 方法 ④，并设置 timer 为 1。然后，调用 restart() 方法 ⑤，并设置 timer 为 DeltaT。最后，调用 update() 方法。

2.3.10 隐藏和显示

在 Spiro 类中，我们添加了一个 toggleTurtles() 方法，用于隐藏和显示所有的 Spiro 对象。

```

# toggle turtle cursor on and off

def toggleTurtles(self):
    for spiro in self.spiros:
        if spiro.t.isvisible():
            spiro.t.hideturtle()
        else:
            spiro.t.showturtle()

```

2.3.11 总结


```
saveDrawing()PNG
```

```
# save drawings as PNG files
```

```
def saveDrawing():
```

```
# hide the turtle cursor
```

```
1 turtle.hideturtle()
```

```
# generate unique filenames
```

```
❷ dateStr = (datetime.now()).strftime("%d%b%Y-%H%M%S")
```

```
fileName = 'spiro-' + dateStr
```

```
print('saving drawing to %s.eps/png' % fileName)
```

```
# get the tkinter canvas
```

```
③ canvas = turtle.getcanvas()
```

```
# save the drawing as a postscript image
```

```
4 canvas.postscript(file = fileName + '.eps')
```

```
# use the Pillow module to convert the postscript
image file to PNG
```

```
5 img = Image.open(fileName + '.eps')
```

```
⑥ img.save(fileName + '.png', 'png')
```

```
# show the turtle cursor
```

```
7 turtle.showturtle()
```

① datetime() ② “-- -- -- -- --

"""A simple program that generates a spiro-gram
using turtle module.

It uses tkinter for the UI. ③ ④
tkinter canvas module to draw PostScript
EPS and PNG. ⑤ Pillow module to save
EPS and PNG. ⑥ ⑦

2.3.12 Command Line Arguments

① main() argparse module

```
① parser = argparse.ArgumentParser(description=descStr)
```

```
# add expected arguments
```

```
② parser.add_argument('--  
sparams', nargs=3, dest='sparams', required=False,
```

```
help="The three arguments in para  
ms: R, r, l.")
```

```
# parse args
```

```
③ args = parser.parse_args()
```

① ② --sparams
③

□□□□□□□□□□turtle□□□

set the width of the drawing window to 80 percent of the screen width

❶ turtle.setup(width=0.8)

set the cursor shape to turtle

❷ turtle.shape('turtle')

set the title to Spirographs!

❸ turtle.title("Spirographs!")

add the key handler to save our drawings

❹ turtle.onkey(saveDrawing, "s")

start listening

❺ turtle.listen()

hide the main turtle cursor

❻ turtle.hideturtle()

□❶□□□setup()□□□□□□□□□□80□□□□□□□□□□
□setup□□□□□□□□□□□❷□□□□□□□□□□□□□❸□□
□□□□□□□□□□Spirographs□□□❹□□□□onkey()□
saveDrawing□□□□S□□□□□□□□□□❺□□□□
listen()□□□□□□□□□□□□□❻□□□□□□□□□□

#####

```
# check for any arguments sent to --  
sparams and draw the Spirograph
```

```
❶ if args.sparams:  
❷     params = [float(x) for x in args.sparams]  
     # draw the Spirograph with the given parameters  
     col = (0.0, 0.0, 0.0)  
❸     spiro = Spiro(0, 0, col, *params)  
❹     spiro.draw()  
else:  
     # create the animator object  
❺     spiroAnim = SpiroAnimator(4)  
     # add a key handler to toggle the turtle cursor  
❻     turtle.onkey(spiroAnim.toggleTurtles, "t")  
     # add a key handler to restart the animation  
❼     turtle.onkey(spiroAnim.restart, "space")  
  
# start the turtle main loop  
❽     turtle.mainloop()
```

❶ ##### --sparams #####
"####"#####❷ #####Python
#####a = [2*x for x
in range(1, 5)]###4#####

③ Spiro Python*
④ draw()

⑤ SpiroAnimator
4 4 ⑥
onkey() T
toggleTurtles ⑦ space
⑧ mainloop()
tkinter

2.4

<https://github.com/electronut/pp/blob/master/spirograph/spiro.py>

```
import sys, random, argparse

import numpy as np

import math

import turtle

import random

from PIL import Image

from datetime import datetime

from fractions import gcd

# a class that draws a Spirograph
```

```

class Spiro:
    # constructor
    def __init__(self, xc, yc, col, R, r, l):

        # create the turtle object
        self.t = turtle.Turtle()

        # set the cursor shape
        self.t.shape('turtle')

        # set the step in degrees
        self.step = 5

        # set the drawing complete flag
        self.drawingComplete = False

        # set the parameters
        self.setparams(xc, yc, col, R, r, l)

        # initialize the drawing
        self.restart()

    # set the parameters
    def setparams(self, xc, yc, col, R, r, l):
        # the Spirograph parameters

```

```
self.xc = xc
self.yc = yc
self.R = int(R)
self.r = int(r)
self.l = l
self.col = col

# reduce r/R to its smallest form by dividing with
the GCD
gcdVal = gcd(self.r, self.R)
self.nRot = self.r//gcdVal
# get ratio of radii
self.k = r/float(R)
# set the color
self.t.color(*col)
# store the current angle
self.a = 0

# restart the drawing
def restart(self):
    # set the flag
    self.drawingComplete = False
    # show the turtle
    self.t.showturtle()
```

```

    # go to the first point

    self.t.up()

    R, k, l = self.R, self.k, self.l

    a = 0.0

    x = R*((1-k)*math.cos(a) + l*k*math.cos((1-k)*a/k))
    y = R*((1-k)*math.sin(a) - l*k*math.sin((1-k)*a/k))

    self.t.setpos(self.xc + x, self.yc + y)

    self.t.down()


# draw the whole thing
def draw(self):

    # draw the rest of the points

    R, k, l = self.R, self.k, self.l

    for i in range(0, 360*self.nRot + 1, self.step):

        a = math.radians(i)

        x = R*((1-k)*math.cos(a) + l*k*math.cos((1-
k)*a/k))

        y = R*((1-k)*math.sin(a) - l*k*math.sin((1-
k)*a/k))

        self.t.setpos(self.xc + x, self.yc + y)

        # drawing is now done so hide the turtle cursor

        self.t.hideturtle()

```



```

# update by one step
def update(self):
    # skip the rest of the steps if done
    if self.drawingComplete:
        return

    # increment the angle
    self.a += self.step

    # draw a step
    R, k, l = self.R, self.k, self.l

    # set the angle
    a = math.radians(self.a)
    x = self.R*((1-k)*math.cos(a) + l*k*math.cos((1-
k)*a/k))
    y = self.R*((1-k)*math.sin(a) - l*k*math.sin((1-
k)*a/k))

    self.t.setpos(self.xc + x, self.yc + y)

    # if drawing is complete, set the flag
    if self.a >= 360*self.nRot:
        self.drawingComplete = True

        # drawing is now done so hide the turtle cursor
        self.t.hideturtle()

# clear everything
def clear(self):

```

```
        self.t.clear()

# a class for animating Spirographs
class SpiroAnimator:
    # constructor
    def __init__(self, N):
        # set the timer value in milliseconds
        self.deltaT = 10

        # get the window dimensions
        self.width = turtle.window_width()
        self.height = turtle.window_height()

        # create the Spiro objects
        self.spiros = []

        for i in range(N):
            # generate random parameters
            rparams = self.genRandomParams()

            # set the spiro parameters
            spiro = Spiro(*rparams)

            self.spiros.append(spiro)

        # call timer
        turtle.ontimer(self.update, self.deltaT)
```

```

# restart spiro drawing
def restart(self):
    for spiro in self.spiros:
        # clear
        spiro.clear()

        # generate random parameters
        rparams = self.genRandomParams()

        # set the spiro parameters
        spiro.setparams(*rparams)

        # restart drawing
        spiro.restart()

# generate random parameters
def genRandomParams(self):
    width, height = self.width, self.height
    R = random.randint(50, min(width, height)//2)
    r = random.randint(10, 9*R//10)
    l = random.uniform(0.1, 0.9)
    xc = random.randint(-width//2, width//2)
    yc = random.randint(-height//2, height//2)
    col = (random.random(),
           random.random(),
           random.random())

```

```
    return (xc, yc, col, R, r, l)
```

```
def update(self):  
    # update all spiros  
    nComplete = 0  
    for spiro in self.spiros:  
        # update  
        spiro.update()  
        # count completed spiros  
        if spiro.drawingComplete:  
            nComplete += 1  
    # restart if all spiros are complete  
    if nComplete == len(self.spiros):  
        self.restart()  
    # call the timer  
    turtle.ontimer(self.update, self.deltaT)  
  
# toggle turtle cursor on and off  
def toggleTurtles(self):  
    for spiro in self.spiros:  
        if spiro.t.isvisible():  
            spiro.t.hideturtle()
```

```

        else:

            spiro.t.showturtle()

# save drawings as PNG files
def saveDrawing():

    # hide the turtle cursor
    turtle.hideturtle()

    # generate unique filenames
    dateStr = (datetime.now()).strftime("%d%b%Y-%H%M%S")
    fileName = 'spiro-' + dateStr
    print('saving drawing to %s.eps/png' % fileName)

    # get the tkinter canvas
    canvas = turtle.getcanvas()

    # save the drawing as a postscript image
    canvas.postscript(file = fileName + '.eps')

    # use the Pillow module to convert the poscript image f
    ile to PNG
    img = Image.open(fileName + '.eps')
    img.save(fileName + '.png', 'png')

    # show the turtle cursor
    turtle.showturtle()

# main() function

```

```

def main():

    # use sys.argv if needed

    print('generating spirograph...')

    # create parser

    descStr = """This program draws Spirographs using the T
urtle module.

    When run with no arguments, this program draws random S
pirographs.

    Terminology:

    R: radius of outer circle
    r: radius of inner circle
    l: ratio of hole distance to r
    """

    parser = argparse.ArgumentParser(description=descStr)

    # add expected arguments

    parser.add_argument('--
sparams', nargs=3, dest='sparams', required=False,
                        help="The three arguments in sparam
s: R, r, l.")

    # parse args

```

```
args = parser.parse_args()

# set the width of the drawing window to 80 percent of
the screen width
turtle.setup(width=0.8)

# set the cursor shape to turtle
turtle.shape('turtle')

# set the title to Spirographs!
turtle.title("Spirographs!")

# add the key handler to save our drawings
turtle.onkey(saveDrawing, "s")

# start listening
turtle.listen()

# hide the main turtle cursor
turtle.hideturtle()

# check for any arguments sent to --
sparams and draw the Spirograph
if args.sparams:
    params = [float(x) for x in args.sparams]
```

```

        # draw the Spirograph with the given parameters
        col = (0.0, 0.0, 0.0)
        spiro = Spiro(0, 0, col, *params)
        spiro.draw()
    else:
        # create the animator object
        spiroAnim = SpiroAnimator(4)
        # add a key handler to toggle the turtle cursor
        turtle.onkey(spiroAnim.toggleTurtles, "t")
        # add a key handler to restart the animation
        turtle.onkey(spiroAnim.restart, "space")

# start the turtle main loop
turtle.mainloop()

# call main
if __name__ == '__main__':
    main()

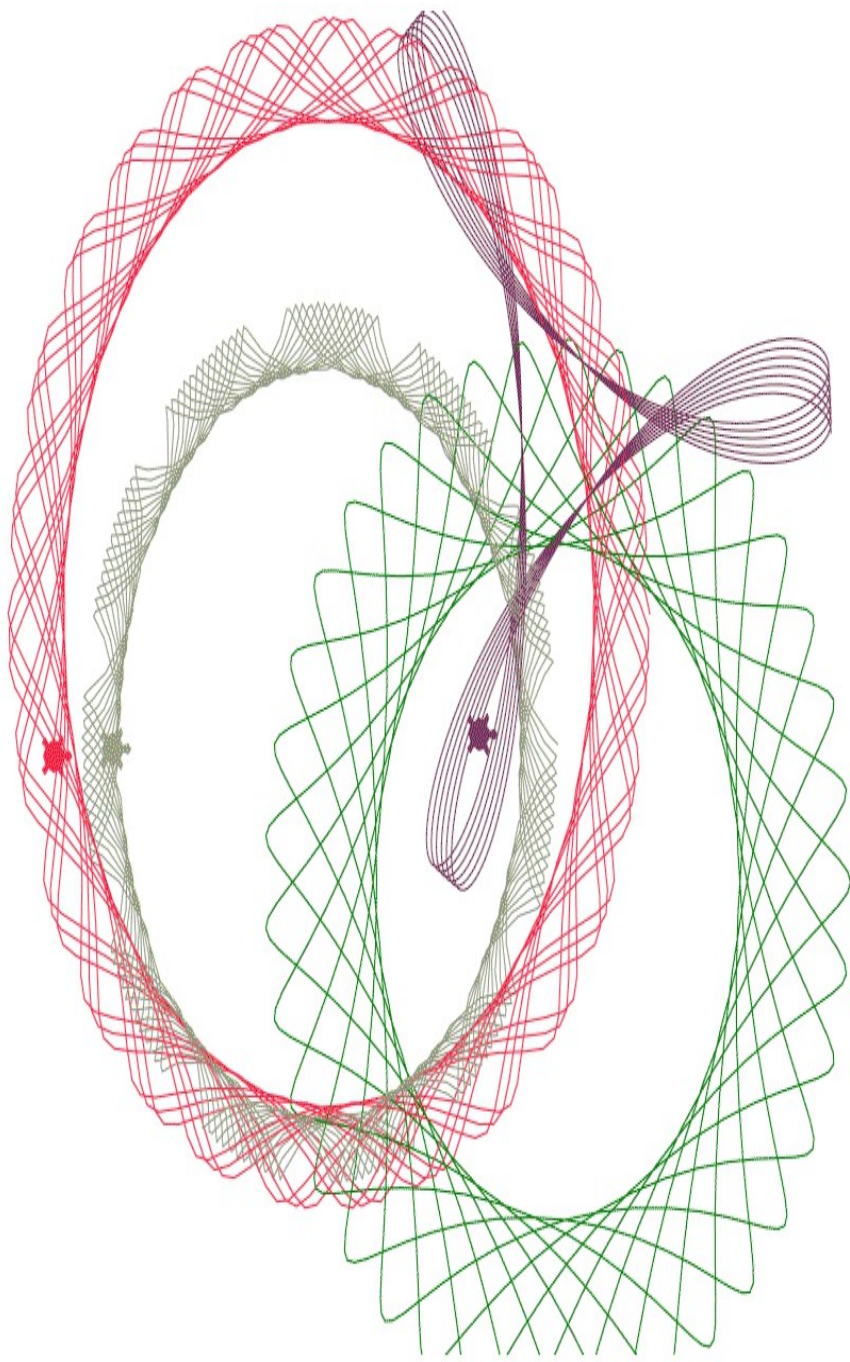
```

2.5 □□□□□□□□

□□□□□□□□□□

\$ **python spiro.py**

spiro.py 2-5 S

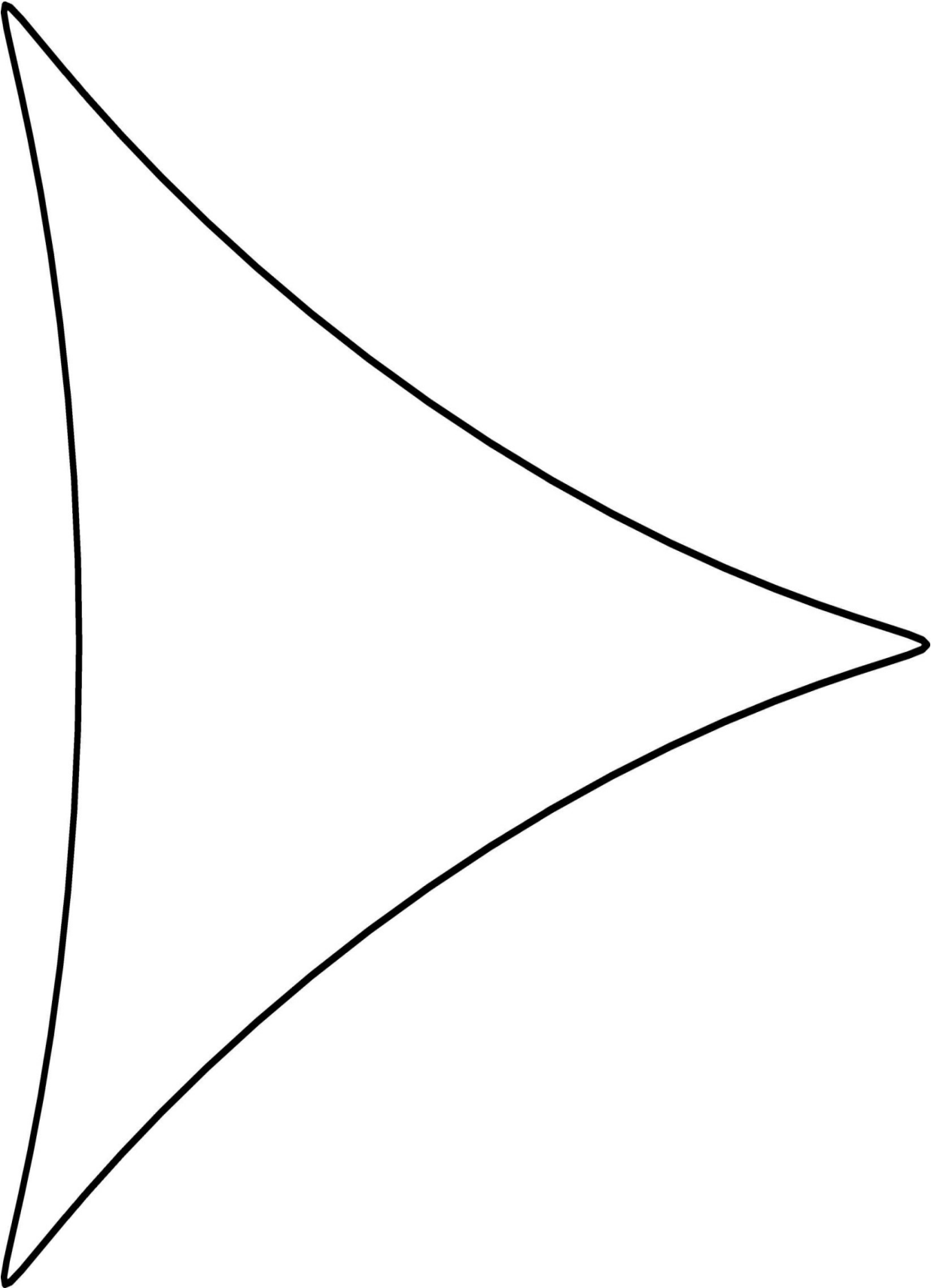


2-5 spiro.py

□ □

```
$ python spiro.py --sparams 300 100 0.9
```

[illegible]



2-6 用turtle模块实现spiro.py

2.6 实现

```
import turtle
t = turtle.Turtle()
t.speed(13)
t.penup()
```

2.7 实现

```
t.pendown()
```

```
1 t.forward(100)
  t.left(90)
```

```
2 t.forward(100)
  t.left(90)
  turtle.setheading(0)
  t.forward(100)
```

```
3 def kochSF(x1, y1, x2, y2, t):
    # recursive Koch snowflake
```

```
# recursive Koch snowflake
```

```
def kochSF(x1, y1, x2, y2, t):
```

```
    # compute intermediate points p2, p3
```

```
    if segment_length > 10:
```

```
        # recursively generate child segments
```

```

# flake #1
kochSF(x1, y1, p1[0], p1[1], t)

# flake #2
kochSF(p1[0], p1[1], p2[0], p2[1], t)

# flake #3
kochSF(p2[0], p2[1], p3[0], p3[1], t)

# flake #4
kochSF(p3[0], p3[1], x2, y2, t)

else:

    # draw

    # ...

```

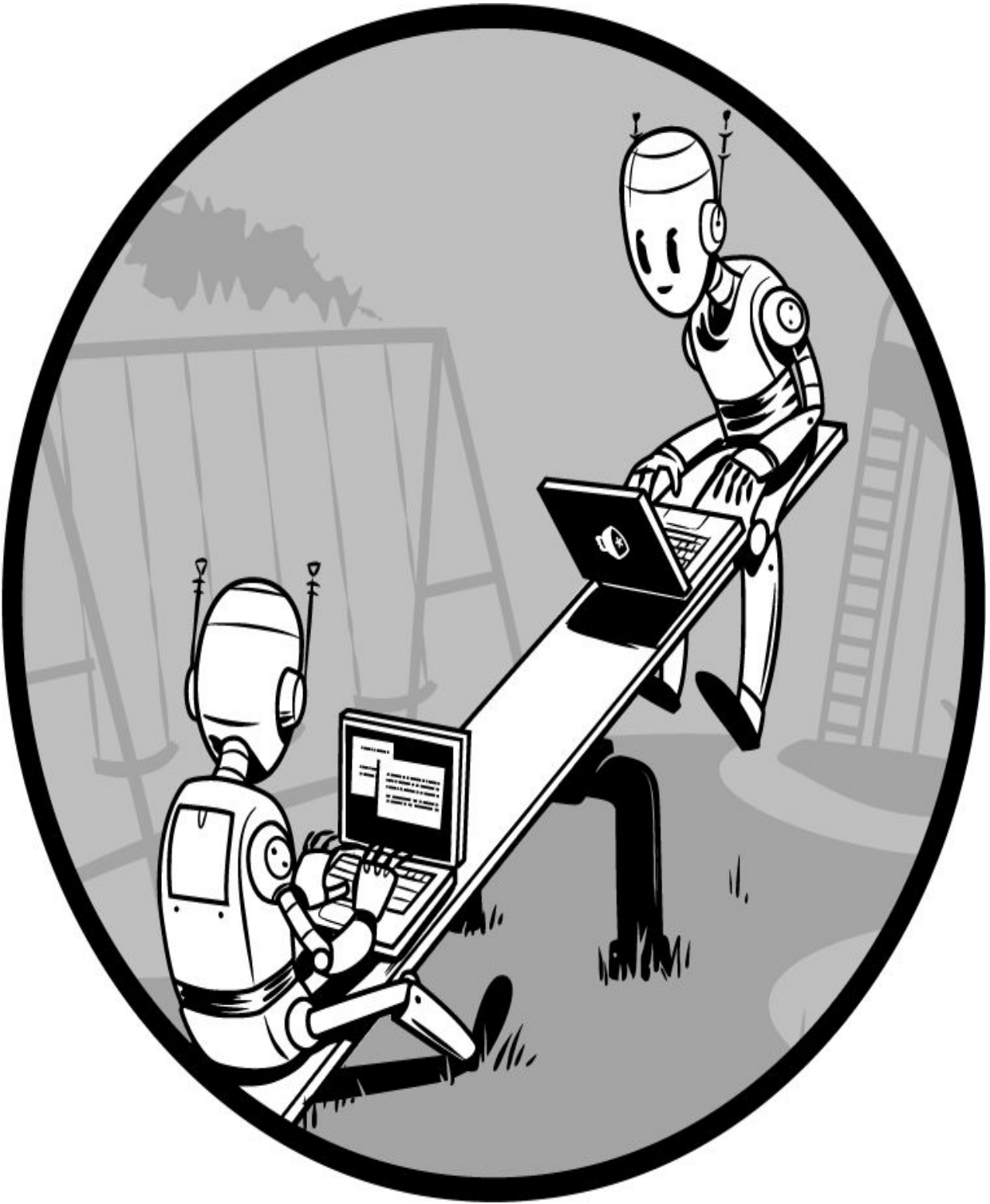
<http://electronut.in/koch-snowflake-and-the-thue-morse-sequence/>

[\[1\] http://en.wikipedia.org/wiki/Spirograph/](http://en.wikipedia.org/wiki/Spirograph/)



“□□□□□□□□□□□□□□□□”

——□□□□□□□□



3 Conway



Conway John
Conway

$N \times N$ Conway

-
-
-

Conway ON OFF
Conway 4 “ ” ON OFF

- matplotlib imshow
- matplotlib
- numpy
- %
-

3.1

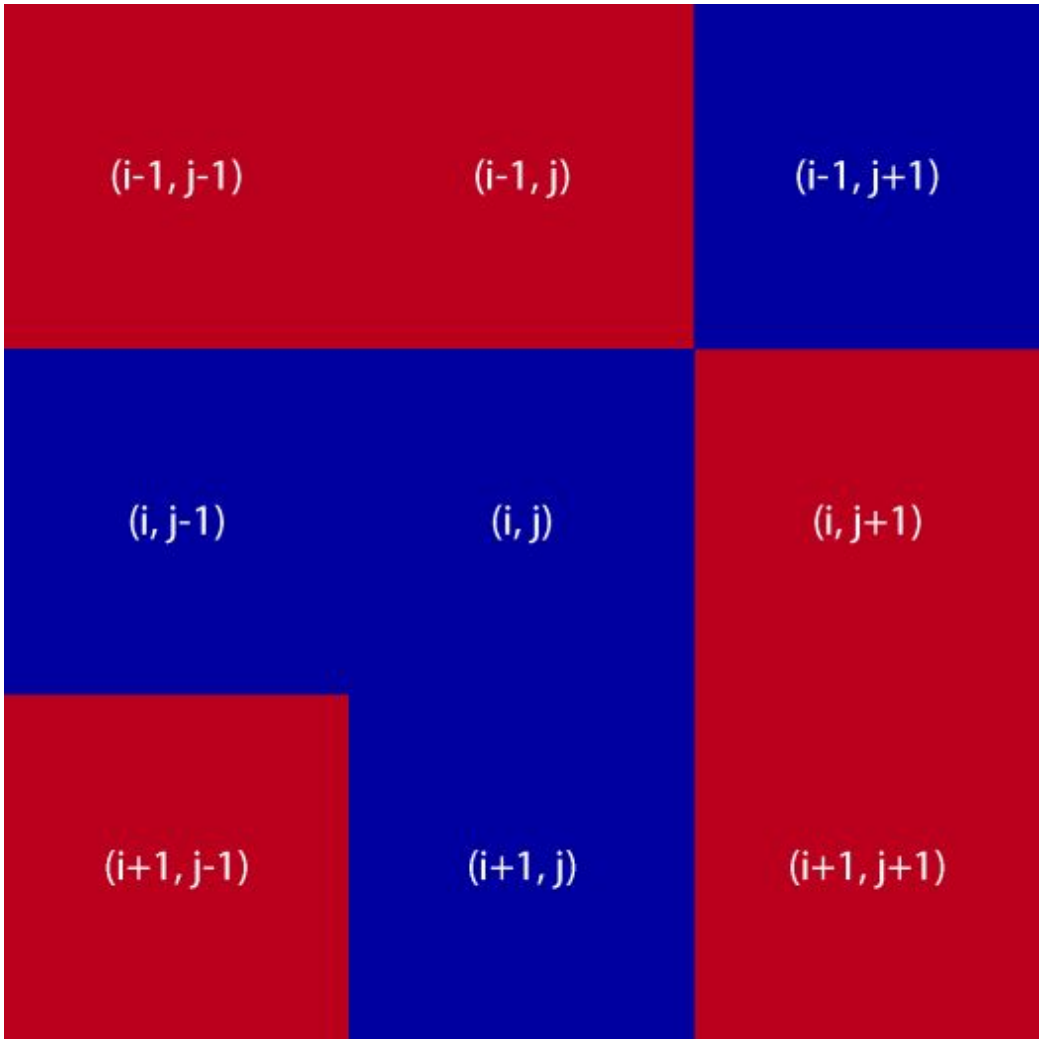
9 8 3-1
(i, j) [i][j] i j
Conway 4

1 ON ON OFF

2 ON 3 ON ON

3 ON 3 ON OFF

4 OFF 3 ON ON

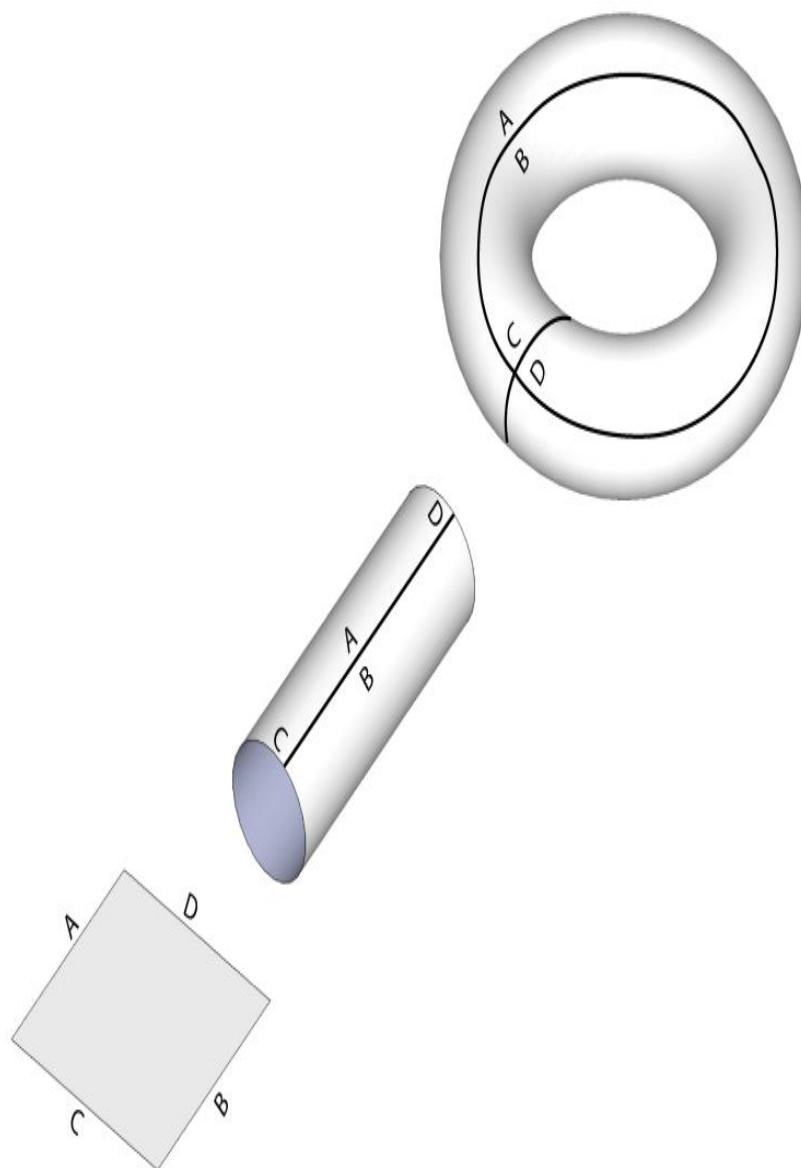


□3-1 8□□□□

[illegible]

Diagram illustrating the instruction format (32 bits):

- Field A (8 bits)
- Field B (8 bits)
- Field C (8 bits)
- Field D (8 bits)



3-2 Pac-Man 4



Pac-Man 4

4

1

2 (i, j)

a (i, j)

b

3.2

numpy matplotlib matplotlib animation matplotlib 1

3.3

Python 3.4

```
>>> import numpy as np
```

```
>>> import matplotlib.pyplot as plt
```

```
>>> import matplotlib.animation as animation
```

[illegible]

3.3.1 〇〇〇〇

```
ONOFF2550
ONOFFmatplotlibimshow()

```

① >>>

```
x = np.array([[0, 0, 255], [255, 255, 0], [0, 255, 0]])
```

```
② >>> plt.imshow(x, interpolation='nearest')
```

```
plt.show()
```

```

❶ numpy.zeros(3×3)
❷ plt.imshow(img, interpolation='nearest')

```

□3-3□□□□□□□□□□

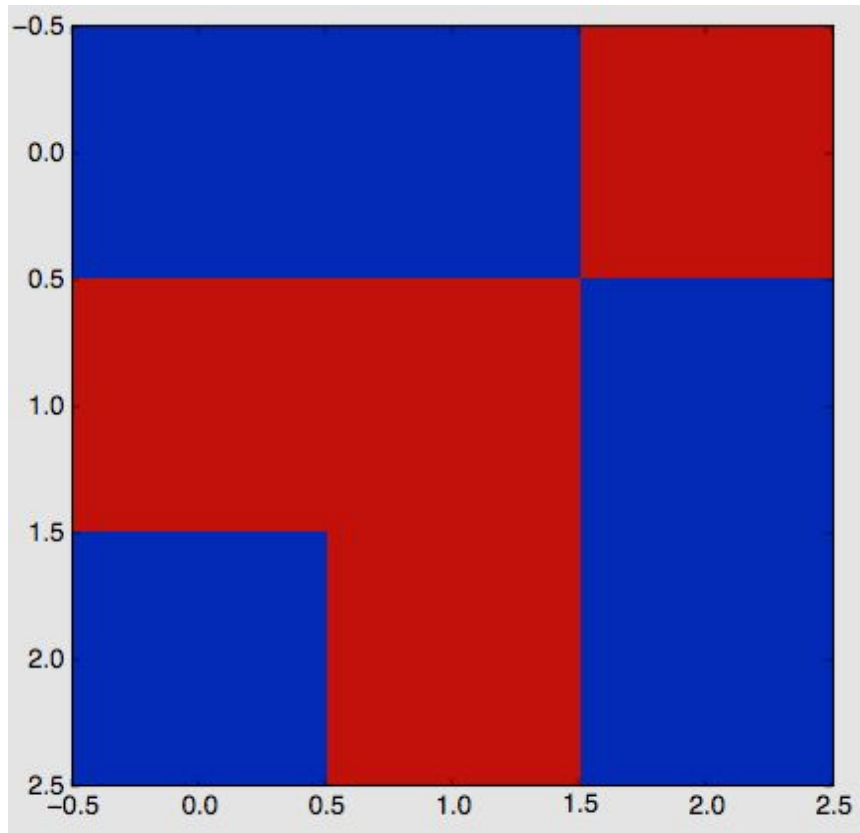


图3-3 棋盘格图

将0置为OFF，将255置为ON，使用imshow()函数

3.3.2 生成图

生成ON和OFF的图，使用numpy.random.choice()函数

使用numpy.random.choice()函数

```
np.random.choice([0, 255], 4*4, p=[0.1, 0.9]).reshape(4, 4)
```

np.zeros(4)

```
array([[255, 255, 255, 255],
       [255, 255, 255, 255],
       [255, 255, 255, 255],
       [255, 255, 255, 0]])
```

```
np.random.choice([0,255],size=(4,4),p=[0.1, 0.9])
# 0.1 is 10% chance of 0, 0.9 is 90% chance of 255
# choice() returns a 1D array of 16 elements
# .reshape() reshapes the array into a 4x4 grid
```

def addGlider(i, j, grid):

"""adds a glider with top left cell at (i, j)"""

```
1 glider = np.array([[0, 0, 255],
                     [255, 0, 255],
                     [0, 255, 255]])

2 grid[i:i+3, j:j+3] = glider

3 grid = np.zeros(N*N).reshape(N, N)

4 addGlider(1, 1, grid)
```

1 3x3 numpy array of zeros

2 3x3 numpy array of ones

이제 `addGlider()` 함수를 `grid`에 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다.

3.3.3 `addGlider()`

`addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다.

```
if j == N-1:
    right = grid[i][0]
else:
    right = grid[i][j+1]
```

이제 `addGlider()` 함수를 `grid`에 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다. `addGlider()` 함수는 `grid`에 `1`을 추가합니다.

```
>>> N = 16

>>> i1 = 14

>>> i2 = 15

>>> (i1+1)%N
15

>>> (i2+1)%N
```

0

grid[i][j] = grid[i][(j+1)%N]
grid[i][j] = grid[i][j]

```
right = grid[i][(j+1)%N]
```

grid[i][j] = N-1
grid[i][j] = (j + 1) % N
grid[i][j] = 0

3.3.4 3D Grid

ON OFF
ON 255
255 ON
ON

```
# apply Conway's rules
```

```
if grid[i, j] == ON:
```

❶ if (total < 2) or (total > 3):

```
    newGrid[i, j] = OFF
```

```
else:
```

```
    if total == 3:
```

❷ newGrid[i, j] = ON

❶ 2 ON 3 ON
ON OFF ❷ OFF 3

main() ① argparse
 ②
 N ③ .mov ④
 ⑤

3.3.6

```
# set grid size

N = 100

if args.N and int(args.N) > 8:
    N = int(args.N)

# set animation update interval
updateInterval = 50
if args.interval:
    updateInterval = int(args.interval)

# declare grid
① grid = np.array([])

# check if "glider" demo flag is specified
if args.glider:
    grid = np.zeros(N*N).reshape(N, N)
    addGlider(1, 1, grid)
```

```
else:
```

```
    # populate grid with random on/off - more off than on
```

```
    grid = randomGrid(N)
```

```
def main():  
    # ❶  
    # ❷
```

```
    # set up the animation
```

```
    ❶ fig, ax = plt.subplots()
```

```
    img = ax.imshow(grid, interpolation='nearest')
```

```
    ❷ ani = animation.FuncAnimation(fig, update, fargs=  
    (img, grid, N, ),
```

```
    frames=10,
```

```
    interval=updateInterval,
```

```
    save_count=50)
```

```
    # number of frames?
```

```
    # set the output file
```

```
    if args.movfile:
```

```
        ani.save(args.movfile, fps=30, extra_args=['-  
vcodec', 'libx264'])
```

```
    plt.show()
```

```
❶ import matplotlib  
animation.FuncAnimation() update() ❷
```

Conway

3.4

<https://github.com/electronut/pp/blob/master/conway/conway.py>

```
import sys, argparse

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation

ON = 255

OFF = 0

vals = [ON, OFF]

def randomGrid(N):

    """returns a grid of NxN random values"""

    return np.random.choice(vals, N*N, p=[0.2, 0.8]).reshape(N, N)

def addGlider(i, j, grid):

    """adds a glider with top-left cell at (i, j)"""
```

```

glider = np.array([[0, 0, 255],
                   [255, 0, 255],
                   [0, 255, 255]])

grid[i:i+3, j:j+3] = glider

def update(frameNum, img, grid, N):
    # copy grid since we require 8 neighbors for calculation
    # and we go line by line
    newGrid = grid.copy()
    for i in range(N):
        for j in range(N):
            # compute 8-
neighbor sum using toroidal boundary conditions
            # x and y wrap around so that the simulation
            # takes place on a toroidal surface
            total = int((grid[i, (j-1)%N] + grid[i, (j+1)%N] +
                        grid[(i-1)%N, j] + grid[(i+1)%N, j] +
                        grid[(i-1)%N, (j-1)%N] + grid[(i-
1)%N, (j+1)%N] +
                        grid[(i+1)%N, (j-
1)%N] + grid[(i+1)%N, (j+1)%N])/255)

            # apply Conway's rules
            if grid[i, j] == 0N:
                if (total < 2) or (total > 3):

```

```

        newGrid[i, j] = OFF
    else:
        if total == 3:
            newGrid[i, j] = ON

    # update data
    img.set_data(newGrid)
    grid[:] = newGrid[:]
    return img,

# main() function
def main():

    # command line arguments are in sys.argv[1], sys.argv[2], .
    ..

    # sys.argv[0] is the script name and can be ignored

    # parse arguments

parser = argparse.ArgumentParser(description="Runs Conway's
    Game of Life

        simulation.")

    # add arguments

    parser.add_argument('--grid-
size', dest='N', required=False)

    parser.add_argument('--mov-
file', dest='movfile', required=False)

```

```

    parser.add_argument('--
interval', dest='interval', required=False)

    parser.add_argument('--
glider', action='store_true', required=False)

    parser.add_argument('--
gosper', action='store_true', required=False)

    args = parser.parse_args()

# set grid size

N = 100

if args.N and int(args.N) > 8:

    N = int(args.N)

# set animation update interval

updateInterval = 50

if args.interval:

    updateInterval = int(args.interval)

# declare grid

grid = np.array([])

# check if "glider" demo flag is specified

if args.glider:

    grid = np.zeros(N*N).reshape(N, N)

    addGlider(1, 1, grid)

```

```

else:
    # populate grid with random on/off - more off than on
    grid = randomGrid(N)

    # set up the animation

    fig, ax = plt.subplots()

    img = ax.imshow(grid, interpolation='nearest')

    ani = animation.FuncAnimation(fig, update, fargs=
    (img, grid, N, ),
                                frames=10,
                                interval=updateInterval,
                                save_count=50)

    # number of frames?

    # set the output file

    if args.movfile:

        ani.save(args.movfile, fps=30, extra_args=['-
        vcodec', 'libx264'])

    plt.show()

# call main

if __name__ == '__main__':
    main()

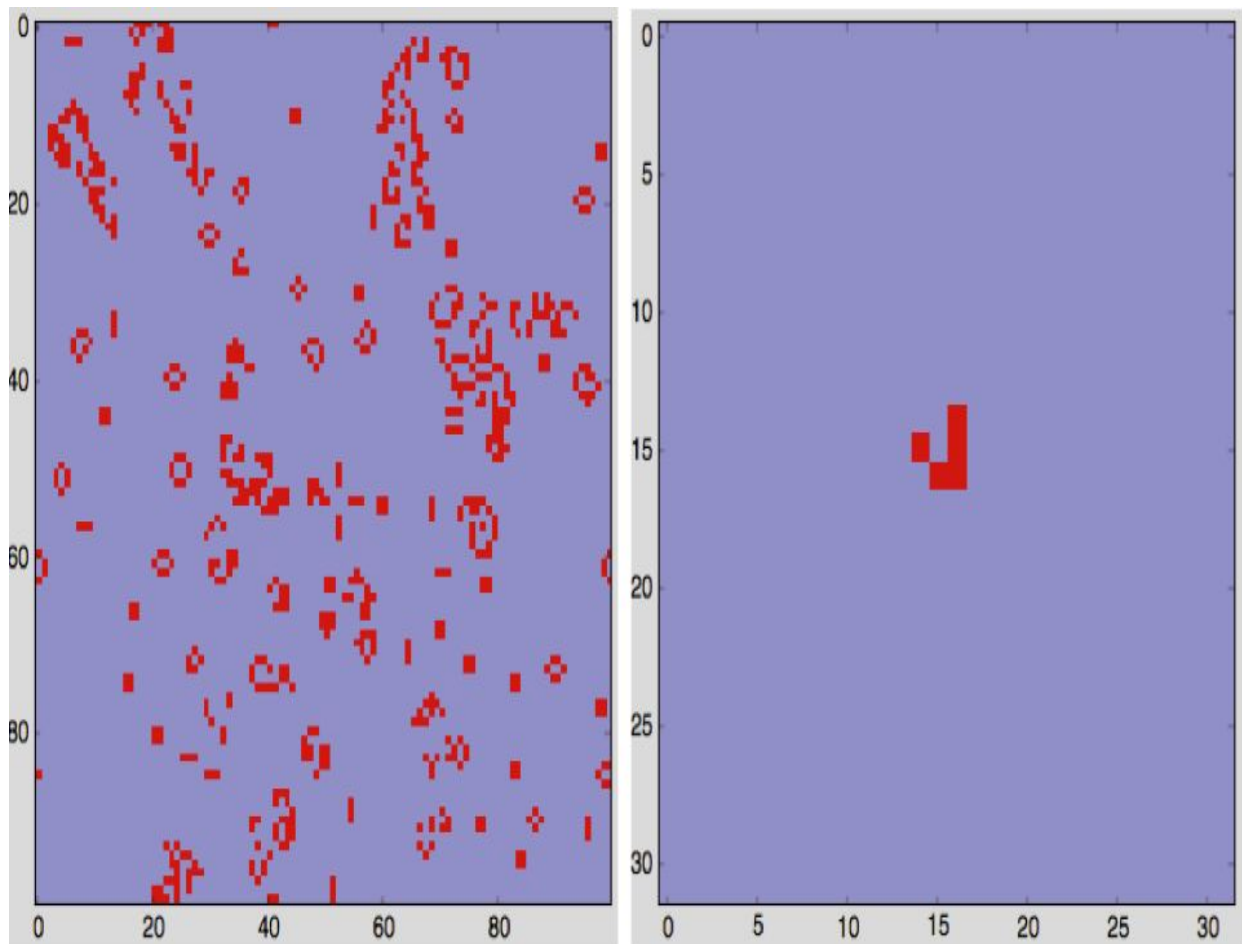
```

3.5

--	--	--	--	--	--	--

```
$ python3 conway.py
```

100×100 50
3-4



(a)

(b)

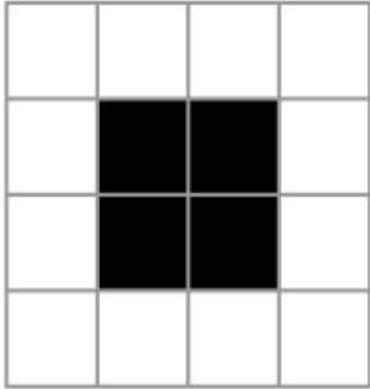
□3-4 □□□□□□□□

3-5

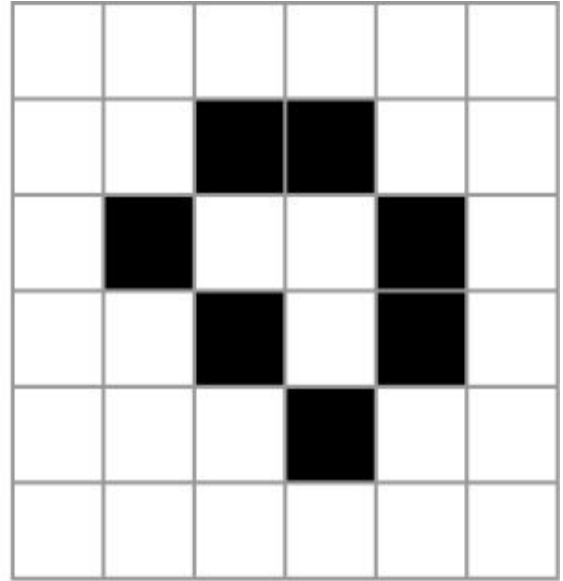
□□□□□□□□□□□□□□□□□□

\$ python conway.py --grid-size 32 --interval 500 --glider

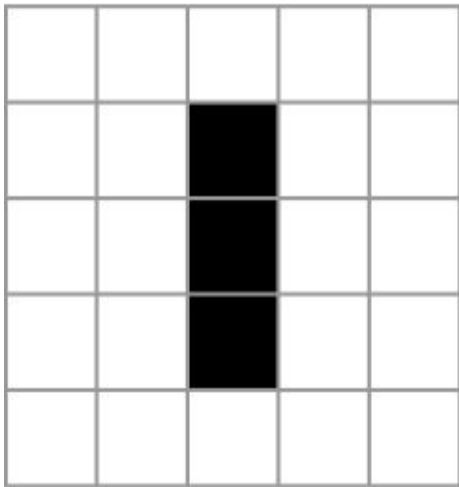
□□□□32×32□□□□□□□500□□□□□□□□□□□□□□□□
□□□□□3-5□□□□□□□



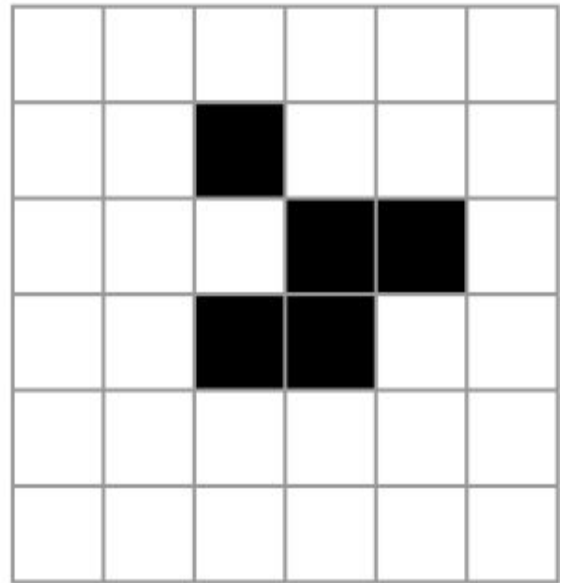
方块



面包



闪光灯（第2阶段）



滑翔机

3.6 练习

在本节中，我们将使用 Conway 的 Game of Life 来演示如何使用 matplotlib 来绘制一个动画。

首先，我们将使用 Conway 的 Game of Life 来生成一个初始状态。然后，我们将使用 matplotlib 来绘制一个动画，展示这个初始状态如何随时间演化。

3.7 练习

在本节中，我们将使用 Conway 的 Game of Life 来演示如何使用 matplotlib 来绘制一个动画。

1. 使用 `addGosperGun()` 方法添加一个 Gosper Gun 到我们的 Conway 游戏中。Gosper Gun 是一个能够生成无限数量的 gliders 的装置。它的名字来源于它的发明者，Gosper。

2. 使用 `readPattern()` 方法读取一个模式文件。这个文件将包含一个初始状态，我们将用它来生成一个动画。

8

0 0 0 255 ...

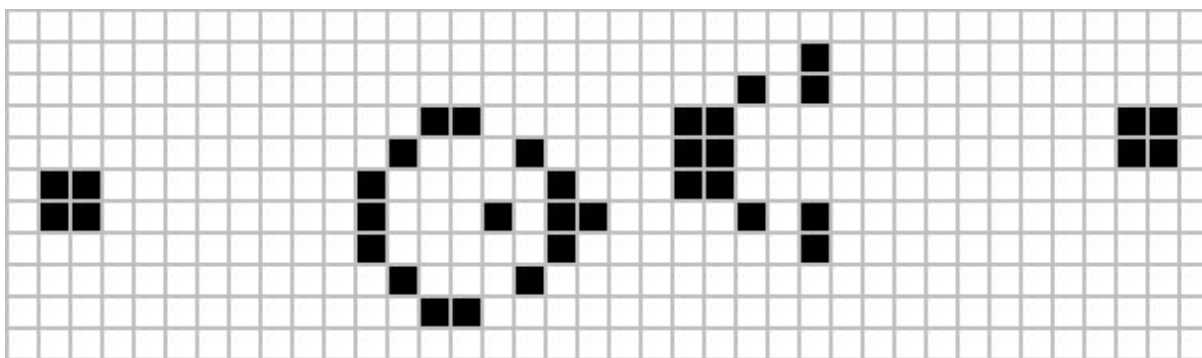


图3-6 图案文件

假设图案文件名为 N ，大小为 $N \times N$ ，每个像素值在 $0 \sim 255$ 之间。
 在 Python 中，使用 `open(file.read())` 读取文件。
 使用 `--pattern-file` 指定图案文件。

第4章 Karplus-Strong 算法

1.1



Hz D 146.83

146.83 Hz

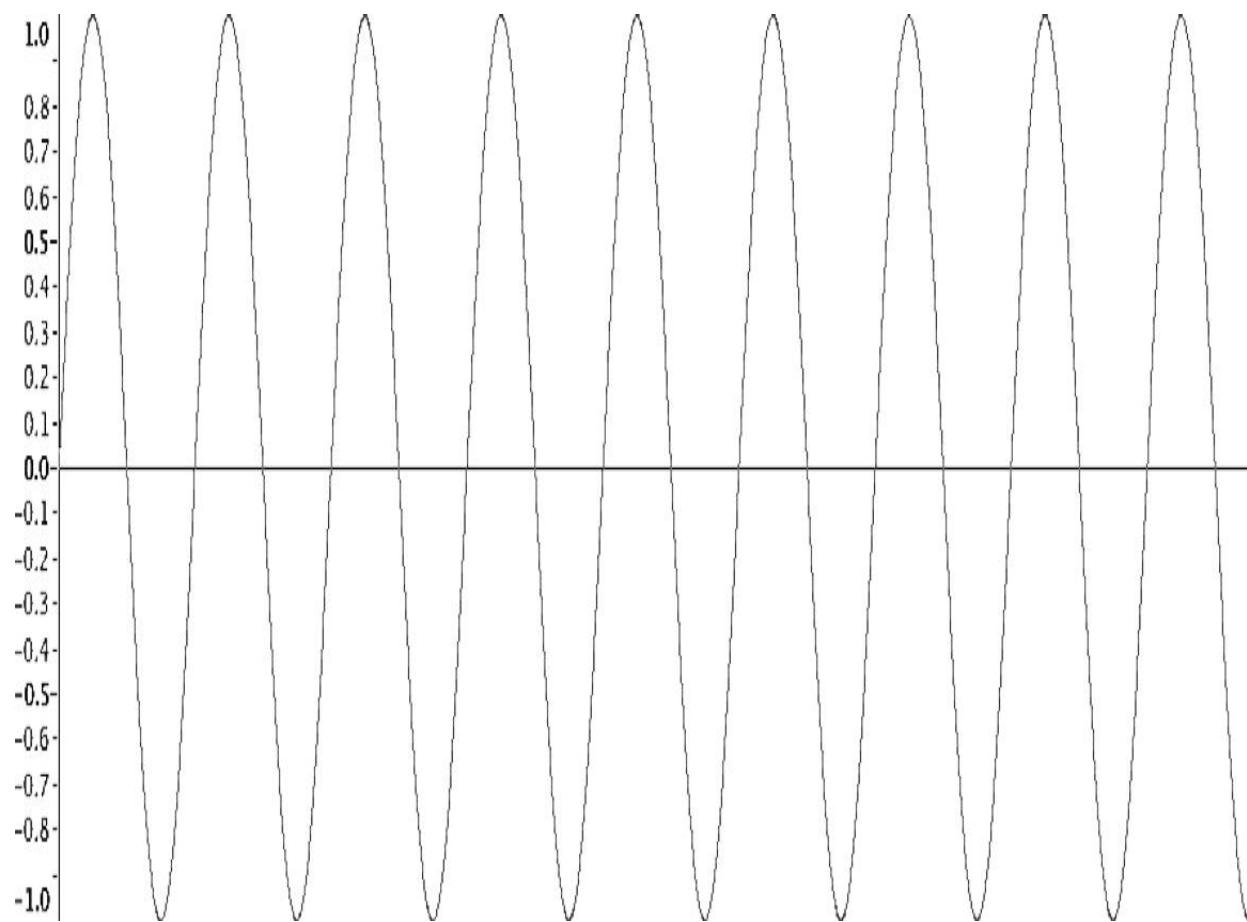
4-1

A grid consisting of two horizontal rows of empty rectangular boxes. The top row contains 15 boxes, and the bottom row also contains 15 boxes, aligned directly beneath the top row.

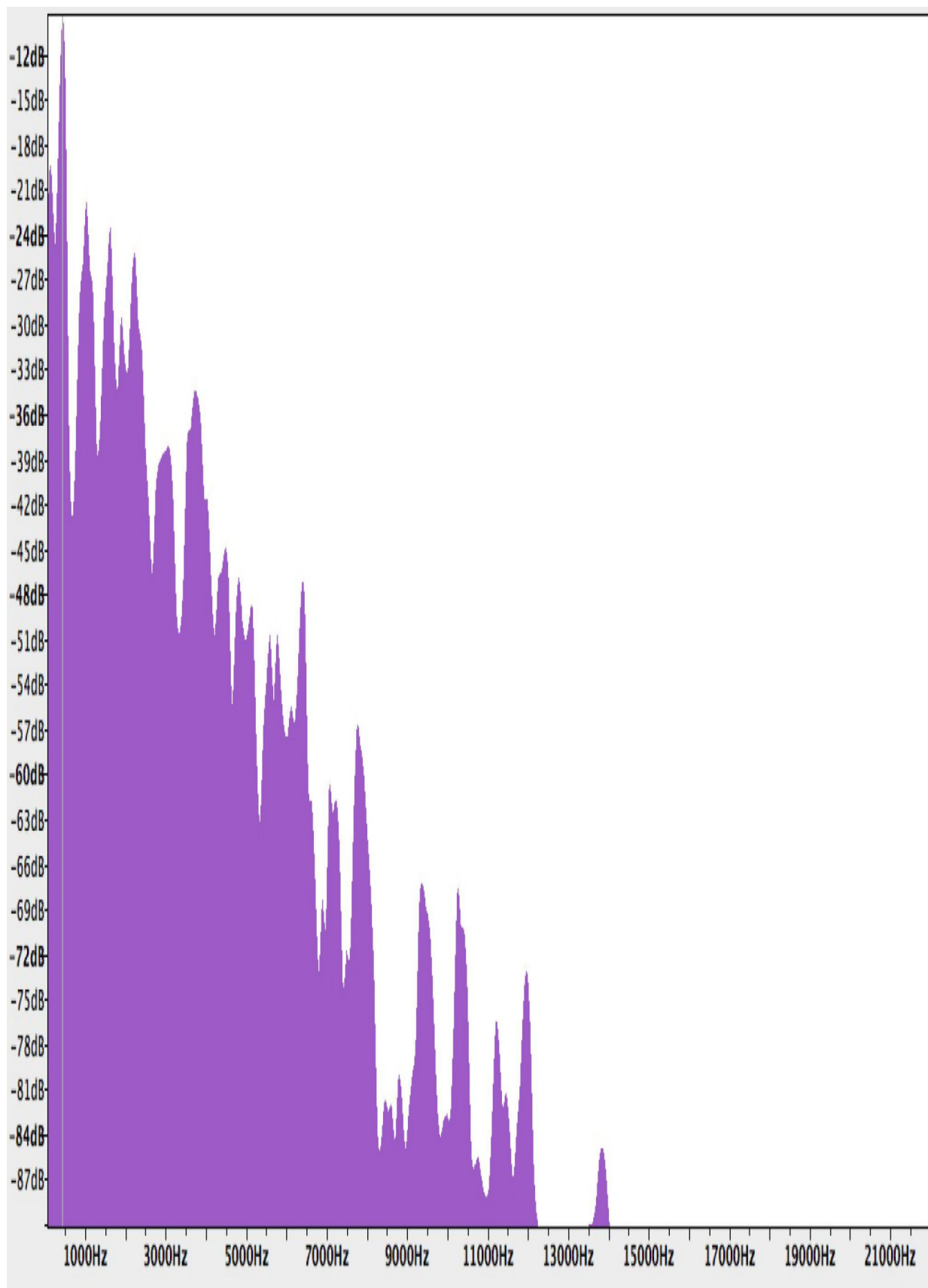
□□□□□□□□□□□□□□□□□□□□□□□□4-2□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□D□□□□□□□□□□
□□□□□□□□146.83□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□
□□

Karplus-Strong 5
WAV

- Python deque
- numpy.ufuncs
- pygame.WAV
- matplotlib
-



□4-1 146.83 Hz□□□□



4-2 用Python实现Karplus-Strong算法

用Python实现Karplus-Strong算法，生成WAV文件。

4.1 算法

Karplus-Strong算法是一种简单的数字滤波算法，用于生成平滑的过渡效果。

该算法的核心思想是将输入信号与一个延迟单元（即4.3.1节中提到的延迟单元）的输出相加，从而实现平滑过渡。

假设输入信号为 $N = S / f$ ，其中 N 为采样率， S 为采样间隔， f 为频率。

该算法的实现步骤如下：

1. 初始化延迟单元，使其输出为0。

4.1.1 延迟单元

延迟单元的实现步骤如下：

1. 初始化延迟单元，使其输出为0。

1. 在 1 秒的时间内，采样 1000 次，得到 1000 个采样值。
 2. 将这 1000 个采样值，按时间顺序排列，得到一个长度为 1000 的数组。
 3. 将这个数组，按时间顺序排列，得到一个长度为 1000 的数组。

1. 在 1 秒的时间内，采样 1000 次，得到 1000 个采样值。
 2. 将这 1000 个采样值，按时间顺序排列，得到一个长度为 1000 的数组。
 3. 将这个数组，按时间顺序排列，得到一个长度为 1000 的数组。

1. 在 1 秒的时间内，采样 1000 次，得到 1000 个采样值。
 2. 将这 1000 个采样值，按时间顺序排列，得到一个长度为 1000 的数组。
 3. 将这个数组，按时间顺序排列，得到一个长度为 1000 的数组。

采样值 1	0.1	-0.2	0.3	0.6	-0.5
采样值 2	-0.2	0.3	0.6	-0.5	-0.199

1. 在 1 秒的时间内，采样 1000 次，得到 1000 个采样值。
 2. 将这 1000 个采样值，按时间顺序排列，得到一个长度为 1000 的数组。
 3. 将这个数组，按时间顺序排列，得到一个长度为 1000 的数组。

4.1.2 采样 WAV 文件

□□□□□□□□WAV□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□WAV□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□16□□□□□WAV□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□44100□□□□CD□□
□□□□□□□□□□Python□□5□□220 Hz□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□

$$A = \sin (2\pi ft)$$

□□□ A □□□□□□□□ f □□□□ t □□□□□□□□□□□□□□□□□□□□
□□□□□

$$A = \sin (2\pi fi / R)$$

□□□□□□□□ i □□□□□□□□ R □□□□□□□□□□□□□□□□□□□□
□□□□□200□□□□□□WAV□□□□

```
import numpy as np

import wave, math

sRate = 44100

nSamples = sRate * 5

❶ x = np.arange(nSamples)/float(sRate)

❷ vals = np.sin(2.0*math.pi*220.0*x)

❸ data = np.array(vals*32767, 'int16').tostring()

file = wave.open('sine220.wav', 'wb')
```

```
④ file.setparams((1, 2, sRate, nSamples, 'NONE', 'uncompressed'))
```

```
file.writeframes(data)
```

```
file.close()
```

① ② numpy
1 sin() numpy
[]

③ [-1 1] 16
④ WAV
16 4-4
sine220.wav Audacity Audacity
220
5 220 Hz

音階の構成は「C + 半音 - 全音 - 半音 + 全音 - 半音」で、CからE、F、G、Bまでの4-1音階（5音階）はKarpplus-Strong音階と呼ばれる。C4は261.6Hz、4は466.2Hz。

4-1 音階の構成

音名	周波数(Hz)
C4	261.6
♭E	311.1
F	349.2
G	392.0
♭B	466.2

4.2 音频

我们使用Python的wave模块来生成WAV文件。我们使用numpy模块来生成Karplus-Strong算法。我们使用Python的deque模块来生成音频。我们使用pygame模块来生成WAV文件。

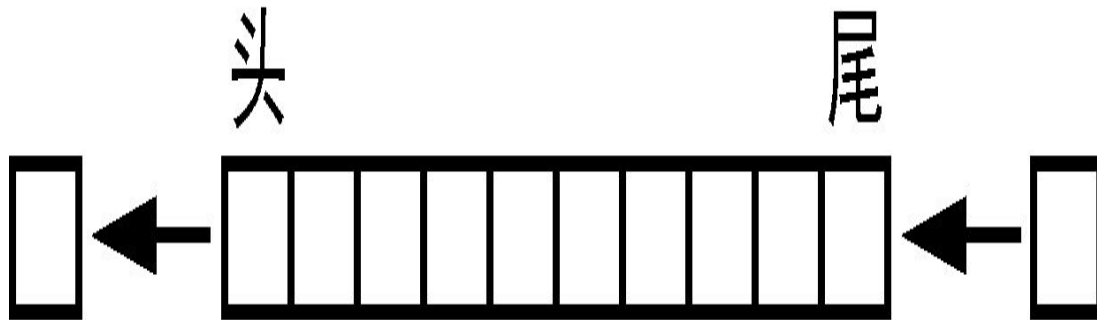
4.3 队列

我们使用Karplus-Strong算法来生成音频。我们使用4.4节

4.3.1 deque队列

我们使用Karplus-Strong算法来生成音频。我们使用Python的deque模块来生成音频。我们使用Python的collections模块来生成音频。我们使用deque模块来生成音频。我们使用4-5节来生成音频。我们使用 $O(1)$ 来生成音频。我们使用deque模块来生成音频。

deque



popleft()

$O(1)$

append()

$O(1)$

4-5 deque

Python deque

```
>>> from collections import deque
```

```
❶ >>> d = deque(range(10))
```

```
>>> print(d)
```

```
deque([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
❷ >>> d.append(-1)
```

```
>>> print(d)
```

```
deque([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -1])
```

```
❸ >>> d.popleft()
```

```
0
```

```
>>> print d
```

```
deque([1, 2, 3, 4, 5, 6, 7, 8, 9, -1])
```

❶ range() deque

❷ deque ❸ deque

4.3.2 Karplus-Strong

deque Karplus-Strong

```
# generate note of given frequency
```

```
def generateNote(freq):
```

```
    nSamples = 44100
```

```
    sampleRate = 44100
```

```
    N = int(sampleRate/freq)
```

```
    # initialize ring buffer
```

```
❶ buf = deque([random.random() - 0.5 for i in range(N)])
```

```
    # initialize samples buffer
```

```
❷ samples = np.array([0]*nSamples, 'float32')
```

```
    for i in range(nSamples):
```

```
❸     samples[i] = buf[0]
```

```

❷    avg = 0.996*0.5*(buf[0] + buf[1])

    buf.append(avg)

    buf.popleft()

# convert samples to 16-bit values and then to a string
# the maximum value is 32767 for 16-bit

❸    samples = np.array(samples*32767, 'int16')

❹    return samples.tostring()

```

❶ 初始化一个空 deque ❷ 遍历所有输入样本

❸ 将 deque 中的样本乘以 32767 ❹ 将 deque 中的样本转换为 16 位整数 ❺ 将 deque 中的样本转换为 16 位整数并存储在 samples 变量中 ❻ 将 samples 变量转换为字符串并返回

4.3.3 WAV

在本节中，我们将使用 Python 的 wave 模块来创建 WAV 文件。

```

def writeWAVE(fname, data):

    # open file

    ❶ file = wave.open(fname, 'wb')

    # WAV file parameters

```

```

nChannels = 1

sampleWidth = 2

frameRate = 44100

nFrames = 44100

# set parameters

❷
file.setparams((nChannels, sampleWidth, frameRate, nFrames,
                'NONE', 'noncompressed'))

❸ file.writeframes(data)

file.close()

```

❶ WAV ❷ 16
 ❸

4.3.4 pygame WAV

Python pygame WAV
 pygame Python
 SDL

NotePlayer

```

# play a WAV file

class NotePlayer:

    # constructor

    def __init__(self):

```

```

❶ pygame.mixer.pre_init(44100, -16, 1, 2048)

    pygame.init()

    # dictionary of notes

❷    self.notes = {}

    # add a note

    def add(self, fileName):

❸        self.notes[fileName] = pygame.mixer.Sound(fileName)

    # play a note

    def play(self, fileName):

        try:

❹            self.notes[fileName].play()

        except:

            print(fileName + ' not found!')

    def playRandom(self):

        """play a random note"""

❺        index = random.randint(0, len(self.notes)-1)

❻        note = list(self.notes.values())[index]

        note.play()

```

```

❶ pygame.mixer.pre_init(44100, -16, 1, 2048)
❷ pygame
NotePlayer.add()
❸
notes

```

play() ④
playRandom()
⑤ randint()[0,4] ⑥

4.3.5 main()

main()

```
parser = argparse.ArgumentParser(description="Generating so  
unds with
```

```
    Karplus String Algorithm")
```

```
    # add arguments
```

```
❶ parser.add_argument('--  
display', action='store_true', required=False)
```

```
    parser.add_argument('--  
play', action='store_true', required=False)
```

```
    parser.add_argument('--  
piano', action='store_true', required=False)
```

```
    args = parser.parse_args()
```

```
    # show plot if flag set
```

```
    if args.display:
```

```
        gShowPlot = True
```

```
        plt.ion()
```

```
    # create note player
```

```

nplayer = NotePlayer()

print('creating notes...')
for name, freq in list(pmNotes.items()):
    fileName = name + '.wav'
    ❷ if not os.path.exists(fileName) or args.display:
        data = generateNote(freq)
        print('creating ' + fileName + '...')
        writeWAVE(fileName, data)
    else:
        print('fileName already created. skipping...')

    # add note to player
    ❸ nplayer.add(name + '.wav')

    # play note if display flag set
    if args.display:
        ❹ nplayer.play(name + '.wav')
        time.sleep(0.5)

# play a random tune
if args.play:

```

```

while True:
    try:
        ⑤ nplayer.playRandom()

        # rest - 1 to 8 beats

        ⑥ rest = np.random.choice([1, 2, 4, 8], 1,
                                   p=[0.15, 0.7, 0.1, 0.05])

        time.sleep(0.25*rest[0])
    except KeyboardInterrupt:
        exit()

```

argparse 模块用于解析命令行参数。
 ① 添加 --display 参数，使用 matplotlib
 Karplus-Strong 算法生成声音。
 matplotlib 模块中的 NotePlayer 类
 generateNote() 方法生成音符。
 pmNotes 列表。

② 检查 WAV 文件是否存在。

WAV 文件 ③ 使用 NotePlayer
 --display ④ 参数。

⑤ 添加 --play 参数，使用 NotePlayer
 playRandom() 方法随机播放。
 numpy.random.choice() 方法。
 ⑥

[illegible]

4.4

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

<https://github.com/electronut/pp/blob/master/karplus/ks.py>

```
import sys, os

import time, random

import wave, argparse, pygame

import numpy as np

from collections import deque

from matplotlib import pyplot as plt

# show plot of algorithm in action?
gShowPlot = False

# notes of a Pentatonic Minor scale
# piano C4-E(b)-F-G-B(b)-C5
pmNotes = {'C4': 262, 'Eb': 311, 'F': 349, 'G': 391, 'Bb': 466}

# write out WAV file

def writeWAVE(fname, data):
```

```

# open file

file = wave.open(fname, 'wb')

# WAV file parameters

nChannels = 1

sampleWidth = 2

frameRate = 44100

nFrames = 44100

# set parameters

file.setparams((nChannels, sampleWidth, frameRate, nFrames,
                'NONE', 'noncompressed'))

file.writeframes(data)

file.close()


# generate note of given frequency
def generateNote(freq):
    nSamples = 44100
    sampleRate = 44100
    N = int(sampleRate/freq)
    # initialize ring buffer
    buf = deque([random.random() - 0.5 for i in range(N)])
    # plot of flag set
    if gShowPlot:

```

```

        axline, = plt.plot(buf)
# initialize samples buffer
samples = np.array([0]*nSamples, 'float32')
for i in range(nSamples):
    samples[i] = buf[0]
    avg = 0.995*0.5*(buf[0] + buf[1])
    buf.append(avg)
    buf.popleft()
# plot of flag set
if gShowPlot:
    if i % 1000 == 0:
        axline.set_ydata(buf)
        plt.draw()

# convert samples to 16-bit values and then to a string
# the maximum value is 32767 for 16-bit
samples = np.array(samples*32767, 'int16')
return samples.tostring()

# play a WAV file
class NotePlayer:
    # constructor
    def __init__(self):
        pygame.mixer.pre_init(44100, -16, 1, 2048)

```

```

    pygame.init()

    # dictionary of notes

    self.notes = {}

# add a note

def add(self, fileName):

    self.notes[fileName] = pygame.mixer.Sound(fileName)

# play a note

def play(self, fileName):

    try:

        self.notes[fileName].play()

    except:

        print(fileName + ' not found!')

def playRandom(self):

    """play a random note"""

    index = random.randint(0, len(self.notes)-1)

    note = list(self.notes.values())[index]

    note.play()


# main() function

def main():

    # declare global var

    global gShowPlot

```

```

parser = argparse.ArgumentParser(description="Generating sounds with

    Karplus String Algorithm")

# add arguments

    parser.add_argument('--
display', action='store_true', required=False)

    parser.add_argument('--
play', action='store_true', required=False)

    parser.add_argument('--
piano', action='store_true', required=False)

args = parser.parse_args()


# show plot if flag set
if args.display:
    gShowPlot = True
    plt.ion()


# create note player
nplayer = NotePlayer()

print('creating notes...')
for name, freq in list(pmNotes.items()):
    fileName = name + '.wav'

```

[illegible]

```

        time.sleep(0.25*rest[0])
    except KeyboardInterrupt:
        exit()

# random piano mode
if args.piano:
    while True:
        for event in pygame.event.get():
            if (event.type == pygame.KEYUP):
                print("key pressed")
                nplayer.playRandom()
                time.sleep(0.5)

# call main
if __name__ == '__main__':
    main()

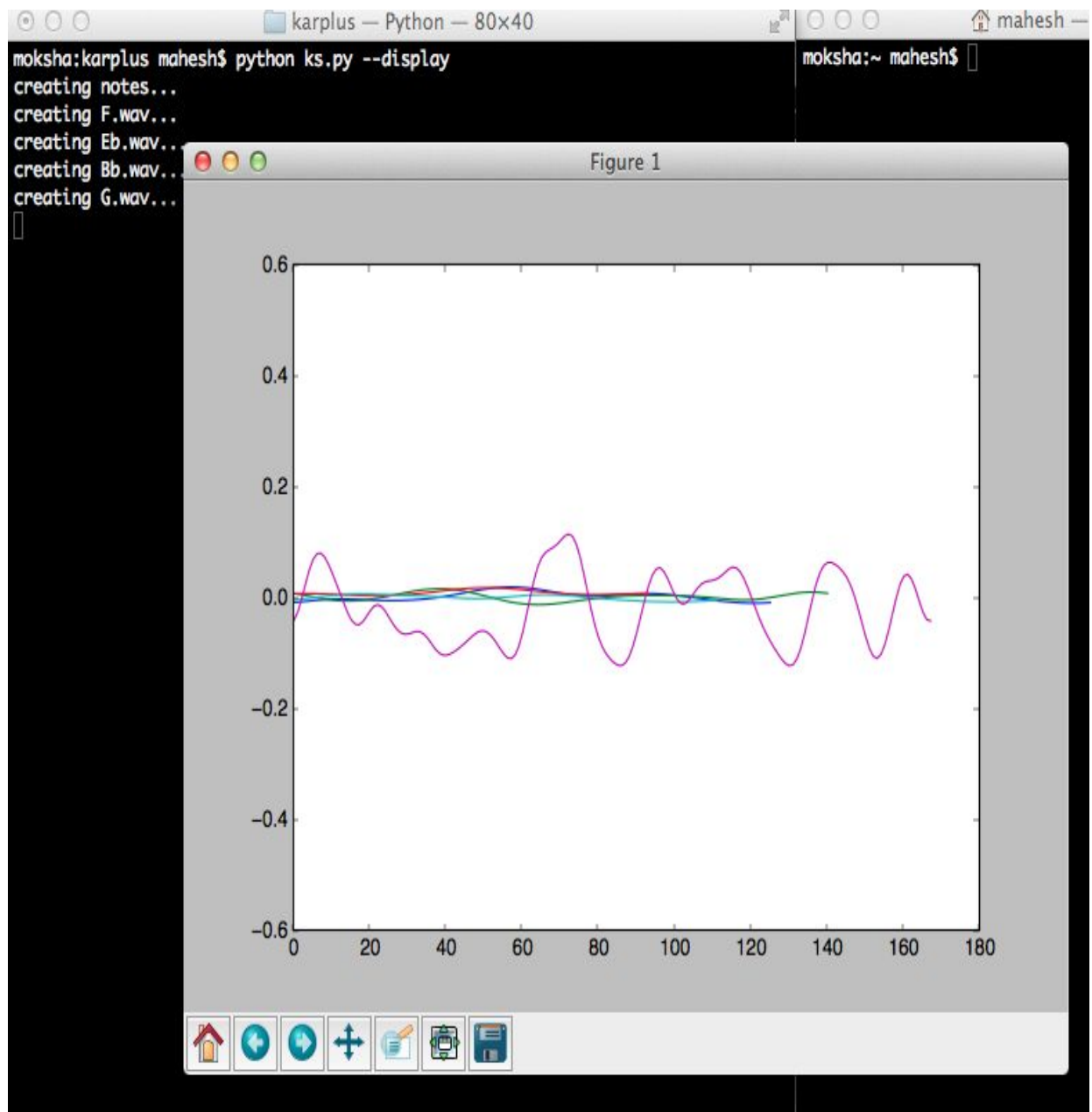
```

4.5 练习

练习 4.5.1: 编写一个程序，使用 pygame 库，实现一个简单的钢琴模式。

\$ python3 ks.py -display

matplotlib Karplus-Strong 4-6



4-6 Karplus-Strong

WAV

\$ python ks.py -play

WAV

4.6 音源

pygame.mixer.Sound()でKarplus-Strong音源を生成し、pygame.mixer.Sound.play()でWAVファイルを読み込み、再生する。

4.7 音源

pygame.mixer.Sound()でKarplus-Strong音源を生成し、pygame.mixer.Sound.play()でWAVファイルを読み込み、再生する。

1 pygame.mixer.Sound()でKarplus-Strong音源を生成し、pygame.mixer.Sound.play()でWAVファイルを読み込み、再生する。

2 pygame.mixer.Sound()でKarplus-Strong音源を生成し、pygame.mixer.Sound.play()でWAVファイルを読み込み、再生する。

3 pygame.mixer.Sound()でKarplus-Strong音源を生成し、pygame.mixer.Sound.play()でWAVファイルを読み込み、再生する。

5 音源



A large grid of empty boxes for writing a story, consisting of 4 rows and 25 columns.

1986年Craig Reynolds年年在加州大学伯克利分校工作，他提出了“Boids”模型，这是第一个模拟鸟群行为的模型。3年后，他出版了《Swarm Intelligence》一书，详细描述了鸟群行为的模拟方法。1992年，他出版了《Swarm Intelligence: A Comprehensive Introduction to the Theory and Applications of Swarm Intelligence》一书。

Reynolds 3 N

5.1

- `scipy.spatial.distance` 距离函数



matplotlib 和 OpenGL 的集成

5.3 图形

本章将介绍如何使用 matplotlib 和 OpenGL 来绘制图形。我们将首先介绍 matplotlib 的基本用法，然后介绍 OpenGL 的基本用法。最后，我们将介绍如何将 matplotlib 和 OpenGL 集成在一起使用。

5.4 图形

5.3.1 环境配置

在本节中，我们将介绍如何配置环境，以便使用 matplotlib 和 OpenGL。我们将首先介绍 matplotlib 的安装，然后介绍 OpenGL 的安装。最后，我们将介绍如何将 matplotlib 和 OpenGL 集成在一起使用。

❶ `import math`

❷ `import numpy as np`

❸ `width, height = 640, 480`

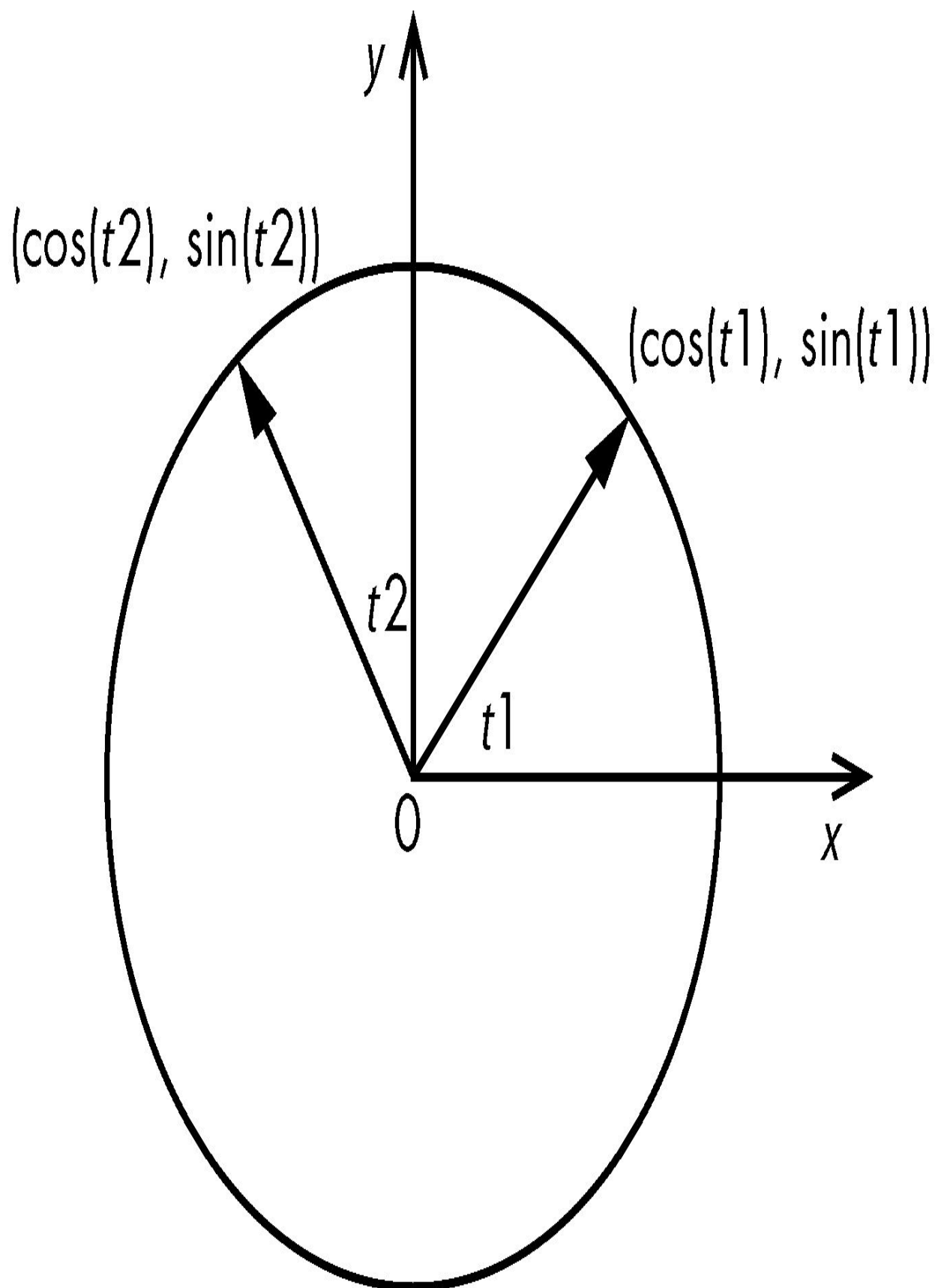
❹ `pos = [width/2.0, height/2.0] + 10*np.random.rand(2*N).reshape(N, 2)`

⑤ `angles = 2*math.pi*np.random.rand(N)`

⑥ `vel = np.array(list(zip(np.sin(angles), np.cos(angles))))`

① `math` ② `numpy`
③ `np` ④ `pos`
`np.random.rand(2 * N)`
`[0, 1]` `2N` `reshape()`
`N` `2` `numpy`
`1×2` `N×2`

`1.0`
`t` `(cos(t), sin(t))`
`1.0` `(0, 0)`
`A` `A`
`5-1`



5-1 zip() 함수 사용하기

⑤ `zip()` 함수는 두 개 이상의 리스트를 인자로 받아서, 길이가 가장 짧은 리스트의 길이만큼 튜플을 생성한다. 예를 들어, `zip([0, 1, 2], [3, 4, 5])`는 `[(0, 3), (1, 4), (2, 5)]`를 생성한다. `zip()` 함수는 Python 3에서부터 사용 가능하다.

```
>>> zip([0, 1, 2], [3, 4, 5])
```

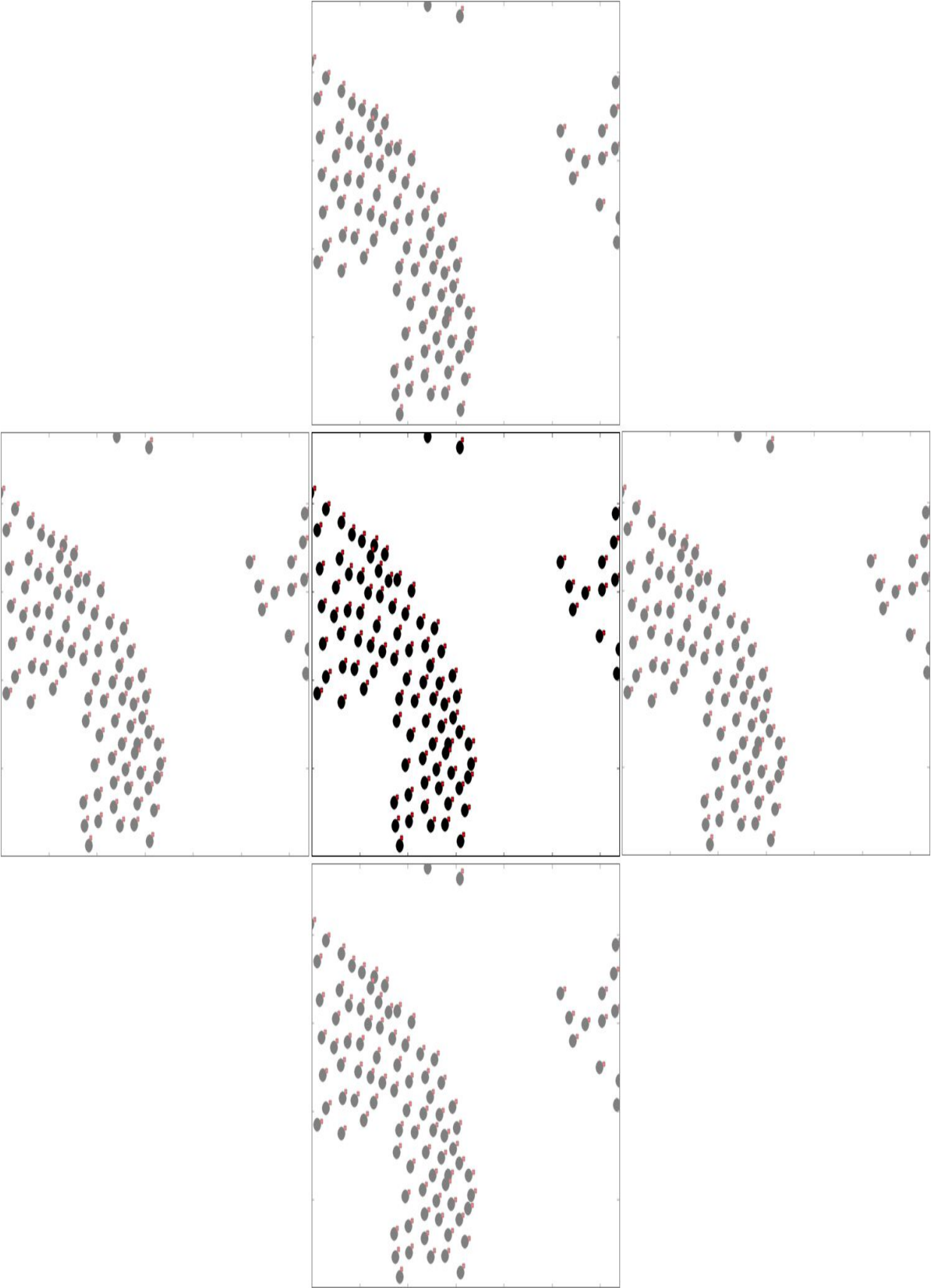
```
[(0, 3), (1, 4), (2, 5)]
```

이제 `zip()` 함수를 사용하여 두 리스트를 결합하여 새로운 리스트를 생성해보자. `zip()` 함수는 두 리스트를 인자로 받아서, 길이가 가장 짧은 리스트의 길이만큼 튜플을 생성한다. 예를 들어, `zip([0, 1, 2], [3, 4, 5])`는 `[(0, 3), (1, 4), (2, 5)]`를 생성한다. `zip()` 함수는 Python 3에서부터 사용 가능하다.

5.3.2 zip() 함수 사용하기

이제 `zip()` 함수를 사용하여 두 리스트를 결합하여 새로운 리스트를 생성해보자. `zip()` 함수는 두 리스트를 인자로 받아서, 길이가 가장 짧은 리스트의 길이만큼 튜플을 생성한다. 예를 들어, `zip([0, 1, 2], [3, 4, 5])`는 `[(0, 3), (1, 4), (2, 5)]`를 생성한다. `zip()` 함수는 Python 3에서부터 사용 가능하다.

이제 `zip()` 함수를 사용하여 두 리스트를 결합하여 새로운 리스트를 생성해보자. `zip()` 함수는 두 리스트를 인자로 받아서, 길이가 가장 짧은 리스트의 길이만큼 튜플을 생성한다. 예를 들어, `zip([0, 1, 2], [3, 4, 5])`는 `[(0, 3), (1, 4), (2, 5)]`를 생성한다. `zip()` 함수는 Python 3에서부터 사용 가능하다.



5-2 边界条件

边界条件的应用

```
def applyBC(self):  
    """apply boundary conditions"""  
    deltaR = 2.0  
    for coord in self.pos:  
        ❶ if coord[0] > width + deltaR:  
            coord[0] = - deltaR  
        if coord[0] < - deltaR:  
            coord[0] = width + deltaR  
        if coord[1] > height + deltaR:  
            coord[1] = - deltaR  
        if coord[1] < - deltaR:  
            coord[1] = height + deltaR
```

❶ 边界条件x轴方向
deltaR 边界条件y轴方向
边界条件x轴方向
边界条件y轴方向

5.3.3 边界条件

边界条件的应用
边界条件的应用

1. 实验目的

通过本实验，使学生掌握利用matplotlib库绘制图形的方法，并理解图5-3所示的几何模型。实验要求如下：
1. 绘制一个大的灰色圆，其圆心为点 P 。
2. 绘制一个小的灰色圆，其圆心为点 H 。
3. 验证公式 $H = P + k \times V$ ，其中 H 为小圆的圆心， V 为小圆的面积， k 为常数。
4. 通过调整参数 k ，观察小圆的位置变化。

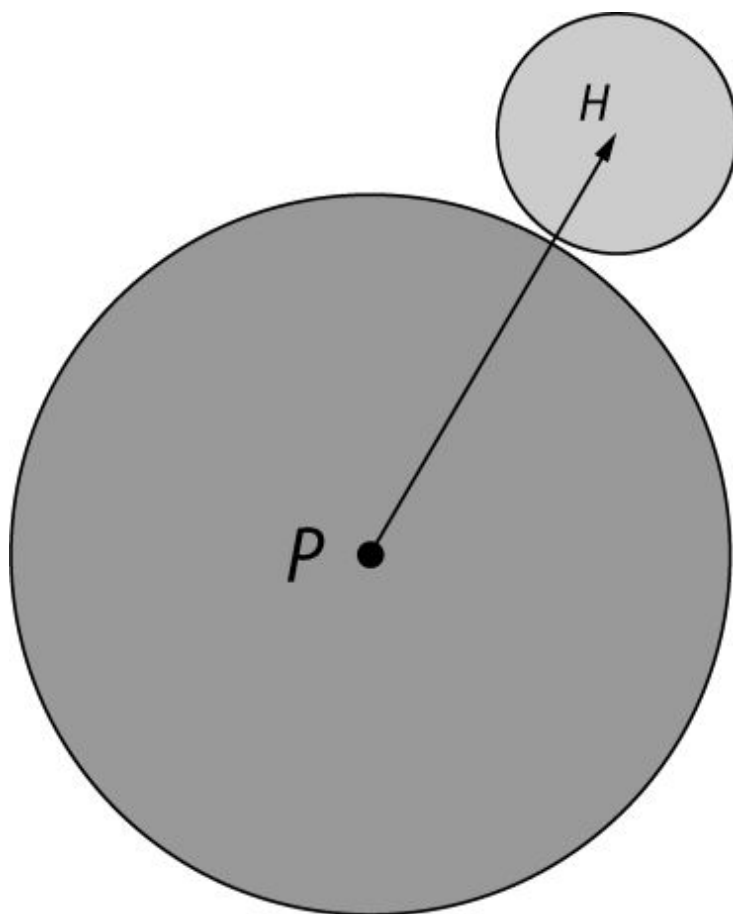


图5-3 几何模型

通过本实验，使学生掌握利用matplotlib库绘制图形的方法，并理解图5-3所示的几何模型。实验要求如下：
1.

```
fig = plt.figure()

ax = plt.axes(xlim=(0, width), ylim=(0, height))
```

❶

```
pts, = ax.plot([], [], markersize=10, c='k', marker='o', l
s='None')
```

❷

```
beak, = ax.plot([], [], markersize=4, c='r', marker='o', l
s='None')
```

❸

```
anim = animation.FuncAnimation(fig, tick, fargs=
(pts, beak, boids),
```

```
interval=50)
```

❶❷ptsbeak

❸

2

❶ `vec = self.pos + 10*self.vel/self.maxVel`

❷ `beak.set_data(vec.reshape(2*self.N)
[:,2], vec.reshape(2*self.N)[1::2])`

❶vel10

❷

`reshapematplotlibset_data[:,2]`

`x[1::2]Y`

5.3.4 □□□□□□

Python 3 “numpy”
numpy

```
import numpy as np

from scipy.spatial.distance import squareform, pdist, cdist


def test2(pos, radius):

    # get distance matrix

    ❶ distMatrix = squareform(pdist(pos))

    # apply threshold

    ❷ D = distMatrix < radius

    # compute velocity

    ❸ vel = pos*D.sum(axis=1).reshape(N, 1) - D.dot(pos)

    return vel
```

```

1 squareform() pdist() scipy

```

```
>>> import numpy as np

>>> from scipy.spatial.distance import squareform, pdist

>>> x = np.array([[0.0, 0.0], [1.0, 1.0], [2.0, 2.0]])

>>> squareform(pdist(x))
```

```
array([[ 0. , 1.41421356, 2.82842712],
       [ 1.41421356, 0. , 1.41421356],
       [ 2.82842712, 1.41421356, 0. ]])
```

squareform() 3×3 矩阵 M_{ij} 和 P_i 、 P_j 都是 3 个元素的列表 ② 矩阵 M 和列表 P 都是 3 个元素的列表

```
>>> squareform(pdist(x)) < 1.4
array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]], dtype=bool)
```

“<” 表示矩阵 M 中的元素 M_{ij} 是否小于 1.4。如果小于，则返回 True，否则返回 False。

③ 使用 `D.sum()` 返回 True，然后使用 `reshape(N, N)` 将结果重塑为 $N \times N$ 的矩阵。这里 `N=1`，所以返回的矩阵是 `D.dot()` 的结果。

test2 和 test1 都是 Python 的模块。timeit 是 Python 的模块。test1 和 test2 都是 test.py 的模块。

```
>>> from timeit import timeit

>>> timeit('test1(pos, 100)', 'from test import test1, N, pos,
```

```
width, height',
```

```
number=100)
```

```
7.880876064300537
```

```
>>>
```

```
timeit('test2(pos, 100)', 'from test import test2, N, pos,  
width, height',
```

```
number=100)
```

```
0.036969900131225586
```

```
#####numpy#####200  
#####
```

```
#####Python#####C#####  
#####numpy#####Python#####  
#####C#####  
#####numpy#####
```

```
#####numpy#####3#####
```

```
def applyRules(self):
```

```
    # apply rule #1: Separation
```

```
    D = distMatrix < 25.0
```

```
    ❶
```

```
    vel = self.pos*D.sum(axis=1).reshape(self.N, 1) - D.dot(sel  
f.pos)
```

```
    ❷
```

```
    self.limit(vel, self.maxRuleVel)
```

```
# distance threshold for alignment (different from separati
```

on)

```
D = distMatrix < 50.0
```

```
# apply rule #2: Alignment
```

```
③ vel2 = D.dot(self.vel)

self.limit(vel2, self.maxRuleVel)

vel += vel2;
```

```
# apply rule #3: Cohesion
```

```
④ vel3 = D.dot(self.pos) - self.pos

self.limit(vel3, self.maxRuleVel)

vel += vel3
```

```
return vel
```

① 初始化所有粒子的位置、速度、加速度等属性。使用 numpy 数组来存储粒子的位置、速度、加速度等属性。

② 初始化所有粒子的邻居列表。使用 numpy 数组来存储粒子的邻居列表。

③ 计算所有粒子的邻居距离。使用 numpy 数组来存储粒子的邻居距离。

numpy 数组的索引从 0 开始，到数组的长度减 1 结束。

④ 计算所有粒子的邻居位置。使用 numpy 数组来存储粒子的邻居位置。

--	--	--	--

5.3.5

[illegible]

```
# add a "button press" event handler
```

```
cid = fig.canvas.mpl_connect('button_press_event', buttonPr  
ess)
```

```

1 mpl_connect() matplotlib
buttonPress()

```

□ □

```
def buttonPress(self, event):
```

```
"""event handler for matplotlib button presses"""
```

```
# left-click to add a boid
```

- ❶ if event.button is 1:

```
② self.pos = np.concatenate((self.pos,
```

```
np.array([[event.xdata, event.ydata]]),
```

axis=0)


```

        # generate a random velocity

❸    angles = 2*math.pi*np.random.rand(1)

v = np.array(list(zip(np.sin(angles), np.cos(angles))))

    self.vel = np.concatenate((self.vel, v), axis=0)

    self.N += 1

```

❶ `event.xdata` and `event.ydata` are the x and y coordinates of the mouse click. `event.button` is the button that was clicked. `1` is the button number.

5.3.6 Scatter Boids

3. In the `buttonPress` function, we will handle the right-click event. We will add a new boid to the simulation. We will use the `UI` class to create a new boid. We will use the `buttonPress` function to handle the right-click event.

```

        # right-click to scatter boids

❶    elif event.button is 3:

        # add scattering velocity

❷    self.vel += 0.1*
        (self.pos - np.array([[event.xdata, event.ydata]]))

```

❶ `event.xdata` and `event.ydata` are the x and y coordinates of the mouse click. `event.button` is the button that was clicked. `3` is the button number.

3

5.3.7

❶
parser = argparse.ArgumentParser(description="Implementing
Craig

Reynolds's Boids...")

add arguments

parser.add_argument('--num-
boids', dest='N', required=False)

args = parser.parse_args()

set the initial number of boids

N = 100

if args.N:

N = int(args.N)

create boids

boids = Boids(N)

main()❶
argparse

5.3.8 Boids

#####Boids#####

```
class Boids:

    """class that represents Boids simulation"""

    def __init__(self, N):

        """initialize the Boid simulation"""

        # initial position and velocities

        ❶
        self.pos = [width/2.0, height/2.0] + 10*np.random.rand(2*N)
        .reshape(N, 2)

        # normalized random velocities

        angles = 2*math.pi*np.random.rand(N)

        self.vel = np.array(list(zip(np.sin(angles), np.cos(angles)
        )))

        self.N = N

        # minimum distance of approach

        self.minDist = 25.0

        # maximum magnitude of velocities calculated by "rules"

        self.maxRuleVel = 0.03

        # maximum magnitude of the final velocity

        self.maxVel = 2.0
```

Boid#####❶#####
#####

5.4 鸟群

鸟群模拟的Python实现

<https://github.com/electronut/pp/blob/master/boids/boids.py>

```
import sys, argparse

import math

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from scipy.spatial.distance import squareform, pdist, cdist

from numpy.linalg import norm


width, height = 640, 480


class Boids:

    """class that represents Boids simulation"""

    def __init__(self, N):

        """initialize the Boid simulation"""

        # initial position and velocities

        self.pos = [width/2.0, height/2.0] + 10*np.random.rand(2*N)

        .reshape(N, 2)

        # normalized random velocities
```

```

        angles = 2*math.pi*np.random.rand(N)

self.vel = np.array(list(zip(np.sin(angles), np.cos(angles)
)))

        self.N = N

        # minimum distance of approach

        self.minDist = 25.0

# maximum magnitude of velocities calculated by "rules"

        self.maxRuleVel = 0.03

        # maximum maginitude of the final velocity

        self.maxVel = 2.0

def tick(self, frameNum, pts, beak):

    """Update the simulation by one time step."""

    # get pairwise distances

    self.distMatrix = squareform(pdist(self.pos))

    # apply rules:

    self.vel += self.applyRules()

    self.limit(self.vel, self.maxVel)

    self.pos += self.vel

    self.applyBC()

    # update data

    pts.set_data(self.pos.reshape(2*self.N)[:2],

```

```

        self.pos.reshape(2*self.N)[1::2])

    vec = self.pos + 10*self.vel/self.maxVel

    beak.set_data(vec.reshape(2*self.N)[::2],
                   vec.reshape(2*self.N)[1::2])

def limitVec(self, vec, maxVal):
    """limit the magnitide of the 2D vector"""
    mag = norm(vec)
    if mag > maxVal:

vec[0], vec[1] = vec[0]*maxVal/mag, vec[1]*maxVal/mag

def limit(self, X, maxVal):
    """limit the magnitide of 2D vectors in array X to ma
xValue"""
    for vec in X:
        self.limitVec(vec, maxVal)

def applyBC(self):
    """apply boundary conditions"""
    deltaR = 2.0
    for coord in self.pos:
        if coord[0] > width + deltaR:

```

```

        coord[0] = - deltaR
    if coord[0] < - deltaR:
        coord[0] = width + deltaR
    if coord[1] > height + deltaR:
        coord[1] = - deltaR
    if coord[1] < - deltaR:
        coord[1] = height + deltaR

def applyRules(self):
    # apply rule #1: Separation

    D = self.distMatrix < 25.0

    vel = self.pos*D.sum(axis=1).reshape(self.N, 1) - D.dot(self.pos)

    self.limit(vel, self.maxRuleVel)

# distance threshold for alignment (different from separation)

    D = self.distMatrix < 50.0

    # apply rule #2: Alignment
    vel2 = D.dot(self.vel)
    self.limit(vel2, self.maxRuleVel)
    vel += vel2;

```



```

    # apply rule #3: Cohesion
    vel3 = D.dot(self.pos) - self.pos
    self.limit(vel3, self.maxRuleVel)
    vel += vel3

    return vel

def buttonPress(self, event):
    """event handler for matplotlib button presses"""
    # left-click to add a boid
    if event.button is 1:
        self.pos = np.concatenate((self.pos,
np.array([[event.xdata, event.ydata]])),
                                axis=0)
        # generate a random velocity
        angles = 2*math.pi*np.random.rand(1)
v = np.array(list(zip(np.sin(angles), np.cos(angles))))
        self.vel = np.concatenate((self.vel, v), axis=0)
        self.N += 1
    # right-click to scatter boids

```

```

        elif event.button is 3:

            # add scattering velocity

            self.vel += 0.1*
            (self.pos - np.array([[event.xdata, event.ydata]]))

def tick(frameNum, pts, beak, boids):

    #print frameNum

    """update function for animation"""

    boids.tick(frameNum, pts, beak)

    return pts, beak

# main() function

def main():

    # use sys.argv if needed

    print('starting boids...')

parser = argparse.ArgumentParser(description="Implementing
Craig

                                Reynold's Boids...")

    # add arguments

    parser.add_argument('--num-
boids', dest='N', required=False)

    args = parser.parse_args()

    # set the initial number of boids

```

```
N = 100

if args.N:
    N = int(args.N)

# create boids
boids = Boids(N)

# set up plot
fig = plt.figure()
ax = plt.axes(xlim=(0, width), ylim=(0, height))

pts, = ax.plot([], [], markersize=10, c='k', marker='o', ls
='None')

beak, = ax.plot([], [], markersize=4, c='r', marker='o', ls
='None')

anim = animation.FuncAnimation(fig, tick, fargs=
(pts, beak, boids),

                                interval=50)

# add a "button press" event handler

cid = fig.canvas.mpl_connect('button_press_event', boids.bu
ttonPress)
```

```
plt.show()
```

```
# call main
```

```
if __name__ == '__main__':
```

```
    main()
```

5.5 鸟群模拟

鸟群模拟

```
$ python3 boids.py
```

鸟群模拟

鸟群模拟 5-4

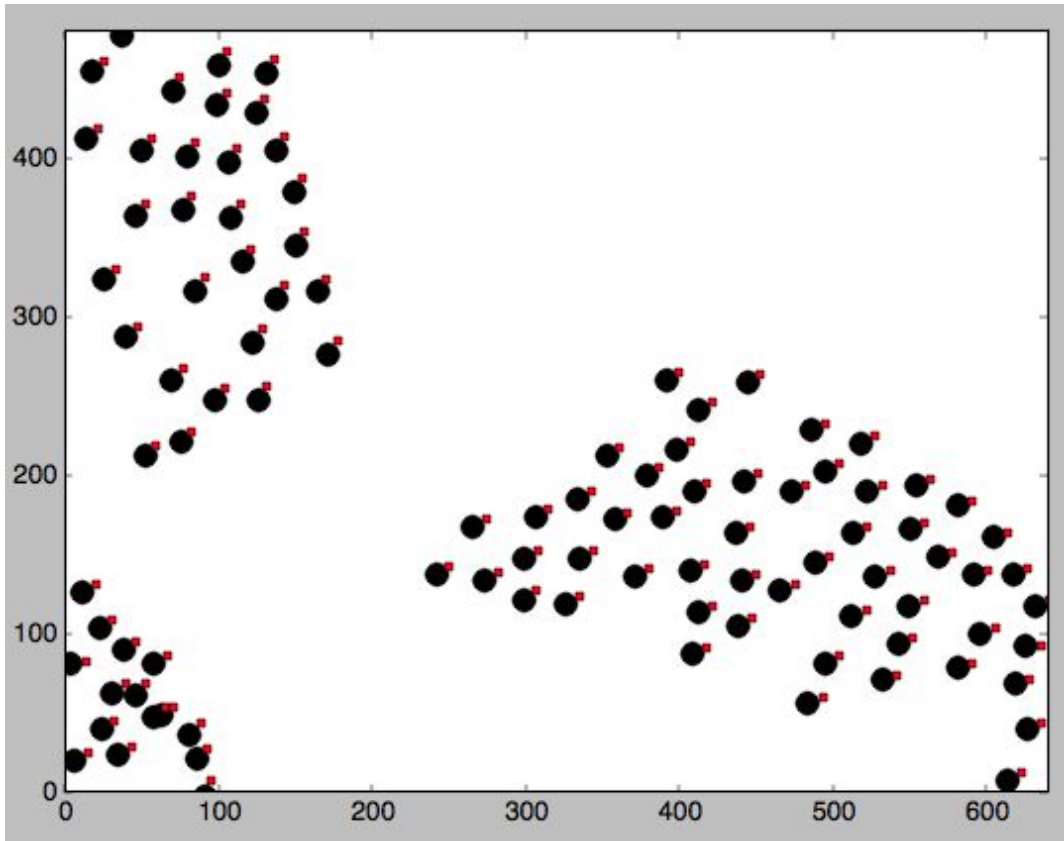


图5-4 散点图

图5-4展示了两个数据簇的散点图。每个数据点由一个黑色圆点和一个红色正方形组成，红色正方形位于黑色圆点的右上方。左侧簇的坐标范围大致在x=0-200, y=0-450，右侧簇的坐标范围大致在x=250-650, y=0-250。

5.6 本章小结

本章介绍了Craig Reynolds的3D粒子系统，以及如何使用numpy和scipy.spatial库进行粒子系统的模拟。本章还介绍了matplotlib库的使用，以及如何使用matplotlib进行数据可视化。本章最后介绍了matplotlib的UI库。

5.7 测试

测试代码通常位于代码的末尾

1 测试代码通常位于代码的末尾
3 测试代码通常位于代码的末尾

```
self.vel += self.applyRules()
```

```
self.vel += self.avoidObstacle()
```

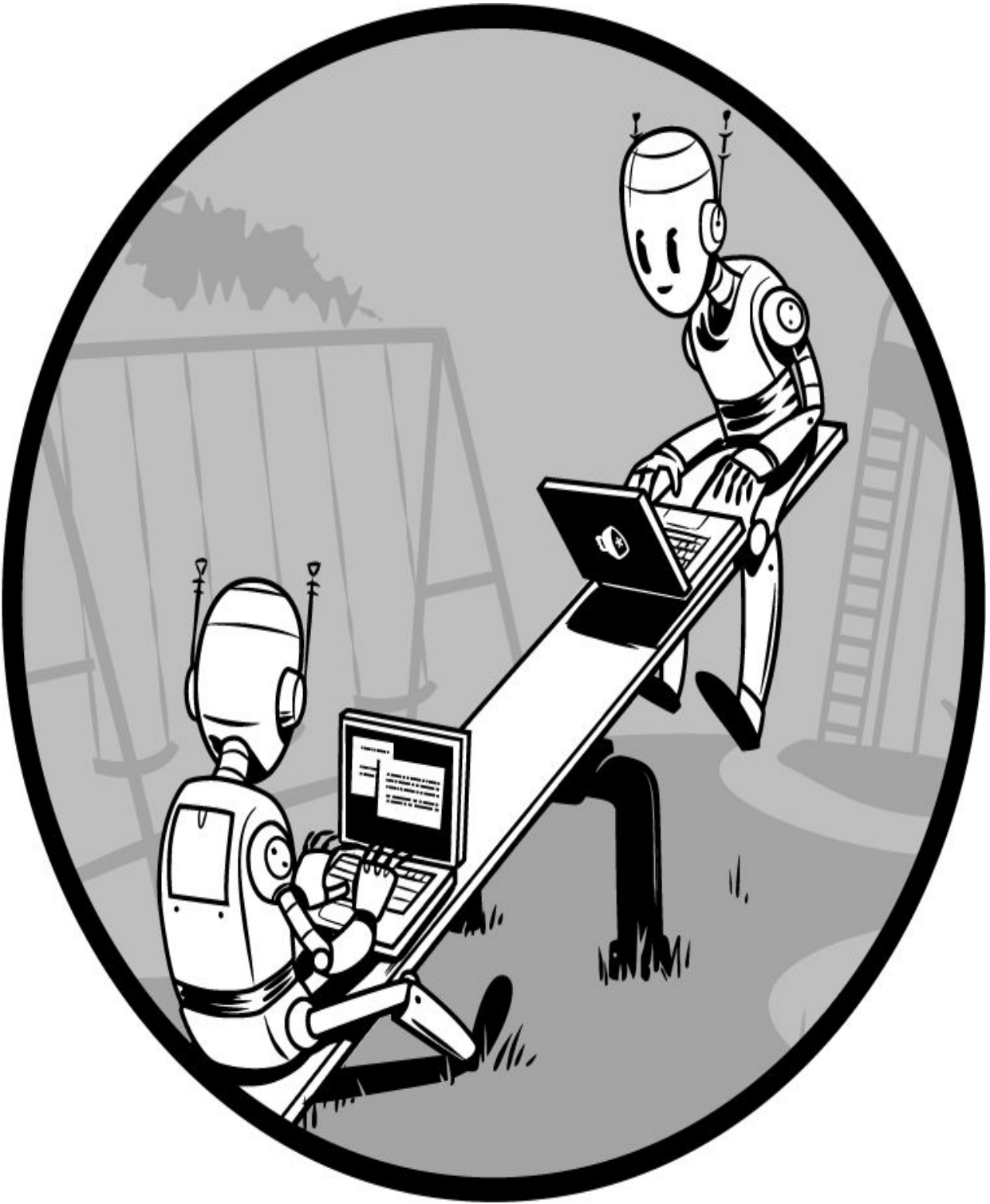
`avoidObstacle()` 测试代码通常位于代码的末尾
`(x, y, R)` 测试代码通常位于代码的末尾
`(x, y)` 测试代码通常位于代码的末尾
`R` 测试代码通常位于代码的末尾
`(x, y, R)` 测试代码通常位于代码的末尾

2 测试代码通常位于代码的末尾
测试代码通常位于代码的末尾
测试代码通常位于代码的末尾

测试代码通常位于代码的末尾

“测试代码通常位于代码的末尾”

——Yogi Berra



第6章 ASCII码



20世纪90年代，随着计算机技术的飞速发展，ASCII码的应用越来越广泛。ASCII码是一种用于表示文本的字符编码标准，它使用7位二进制数来表示128个不同的字符。在6-1节中，我们将详细介绍ASCII码的编码方式和应用。

ASCII码的编码方式非常简单，它使用7位二进制数来表示128个不同的字符。在1960年代，ASCII码被广泛采用，成为计算机系统中的一种标准。20世纪60年代，ASCII码被广泛采用，成为计算机系统中的一种标准。ASCII码的编码方式非常简单，它使用7位二进制数来表示128个不同的字符。

Python是一种高级编程语言，它使用ASCII码来表示文本。在Python中，ASCII码的编码方式非常简单，它使用7位二进制数来表示128个不同的字符。在10-70节中，我们将详细介绍Python的编码方式和应用。

ASCII码的应用非常广泛，它不仅用于表示文本，还用于表示二进制数据。在ASCII码中，每个字符都有一个唯一的二进制表示，这使得ASCII码成为一种非常有效的编码方式。

- ```

 \\\|/
 (o o)
 o00o(-)_o00o_

+ +
+ +
+-----+-----+
+Postdoctoral Researcher | e-mail-->rxj10@psu.edu +
+Materials Research Laboratory | Tel -->(814) 865-9931 +
+Pennsylvania State University | fax -->(814) 863-6734 +
+-----+-----+
+ web page --> http://www.personal.psu.edu/~rxj10 +
+-----+-----+
 || ||
 0000 0000

```

The image displays a highly complex, dense pattern of vertical lines and symbols, resembling a barcode or a highly stylized text representation. The pattern is composed of numerous vertical lines of varying heights and widths, interspersed with various symbols, including asterisks, hash marks, and dots. The overall effect is a dense, textured wall of information, with the symbols and lines arranged in a way that suggests a structured, yet highly complex, data set or a highly stylized text representation. The pattern is dense and fills the entire page, with no visible margins or other content.

## 6.1 1111

ASCII 6-2 ASCII RGB ASCII “ ”

ASCII 6-2 ASCII RGB ASCII “ ”



6-2

8 [0,255] 8 0 255

$M \times N$  M N ASCII [0,255] ASCII

ASCII Courier  
ASCII  
ASCII

“ ”  
ASCII  
ASCII  
ASCII

Courier

ASCII

1

2  $M \times N$

3  $M$

4 ASCII

5 ASCII

## 6.2

Pillow Python  
numpy

## 6.3 ASCII Art

ASCII art is a form of digital art created from printable ASCII characters. It is often used to represent images, text, and other graphical elements in a way that can be displayed in a terminal window or a text-based environment.

6.4

### 6.3.1 ASCII Art

ASCII art is a form of digital art created from printable ASCII characters. It is often used to represent images, text, and other graphical elements in a way that can be displayed in a terminal window or a text-based environment.

# 70 levels of gray

① gscale1 = "\$@B%8&WM#\*oahkbdpqwmZ00QLCJUYXzcvunxrjft/\|  
( )1{ }[ ]? \_+~<>i!lI;:;,\"^`\". "

# 10 levels of gray

② gscale2 = "@%#\*+=-:. " "

① gscale1 70 ② gscale2 10  
10  
Paul Bourke  
Character Representation of Grey Scale  
Images

<http://paulbourke.net/dataformats/asciiart/>

```

open the image and convert to grayscale
❶ image = Image.open(fileName).convert("L")

store the image dimensions

❷ W, H = image.size[0], image.size[1]

compute the tile width

❸ w = W/cols

compute the tile height based on the aspect ratio and scale of the font

❹ h = w/scale

compute the number of rows to use in the final grid

❺ rows = int(H/h)

```

❶ `Image.open()` opens the image file and `Image.convert()` converts the image to grayscale (luminance).

❷ `image.size` returns a tuple of (width, height). `cols` is the number of columns (width divided by 80). `rows` is the number of rows (height divided by `h`).

❸ `scale` is a variable that controls the font size. `h` is the height of the font.

❹ `scale` is a variable that controls the font size. `0.43` is a constant value.

Courier

## 6.3.2

getAverageL()

```
❶ def getAverageL(image):
 # get the image as a numpy array
 ❷ im = np.array(image)
 # get the dimensions
 ❸ w,h = im.shape
 # get the average
 ❹ return np.average(im.reshape(w*h))
```

❶ PIL Image ❷ image  
numpy im  
❸ ❹  
numpy.average()  
numpy.reshape(w\*h)  
numpy.average()

## 6.3.3 ASCII

ASCII

```
an ASCII image is a list of character strings
```

```

❶ aimg = []

generate the list of tile dimensions
for j in range(rows):

❷ y1 = int(j*h)
 y2 = int((j+1)*h)

 # correct the last tile
 if j == rows-1:
 y2 = H

 # append an empty string

❸ aimg.append("")

 for i in range(cols):

 # crop the image to fit the tile

❹ x1 = int(i*w)
 x2 = int((i+1)*w)

 # correct the last tile

❺ if i == cols-1:
 x2 = W

crop the image to extract the tile into another Image object

❻ img = image.crop((x1, y1, x2, y2))

 # get the average luminance

❼ avg = int(getAverageL(img))

```

```
if moreLevels:
```

```
else:
```

```
append the ASCII character to the string
```

[illegible][illegible]

③ ASCII

```

4 x 5 x
y 6 image.crop()
getAverageL() 7 6.3.2
9 [0 255]
[0 9] 10 gscale2
ASCII 8

```



70 10  
ASCII gval

## 6.3.4

argparse

```
parser = argparse.ArgumentParser(description="descStr")

add expected arguments

❶ parser.add_argument('--
file', dest='imgFile', required=True)

❷ parser.add_argument('--
scale', dest='scale', required=False)

❸ parser.add_argument('--
out', dest='outFile', required=False)

❹ parser.add_argument('--
cols', dest='cols', required=False)

❺ parser.add_argument('--
morelevels', dest='moreLevels', action='store_true')
```

❶ ❷  
❸ ❹ ASCII ❺  
--morelevels

## 6.3.5 ASCII

ASCII

```
open a new text file

❶ f = open(outFile, 'w')
```

```

 # write each string in the list to the new file

❷ for row in aimg:

 f.write(row + '\n')

 # clean up

❸ f.close()

```

❶ `open()` `open('image.png', 'r')`

❷ `f.write(row + '\n')` `f.write(row + '\n')` `f.write(row + '\n')` `f.write(row + '\n')` `f.write(row + '\n')`

❸ `f.close()`

## 6.4 ASCII Art

ASCII Art

<https://github.com/electronut/pp/blob/master/ascii/ascii.py>

```

import sys, random, argparse

import numpy as np

import math

from PIL import Image

grayscale level values from:

http://paulbourke.net/dataformats/asciiart/

70 levels of gray

gscale1 = "$@B%8&WM#*oahkbdpqwmZ00QLCJUyXzcvunxrjft/\|()1{}
[]?-_+~<>i!lI;:;,\"^

```

```

 \', "

10 levels of gray

gscale2 = '@%#*+=-:. '

def getAverageL(image):
 """
 Given PIL Image, return average value of grayscale value
 """

 # get image as numpy array
 im = np.array(image)

 # get the dimensions
 w,h = im.shape

 # get the average
 return np.average(im.reshape(w*h))

def covertImageToAscii(fileName, cols, scale, moreLevels):
 """

 Given Image and dimensions (rows, cols), returns an m*n list
 of Images

 """

 # declare globals
 global gscale1, gscale2

```

```

open image and convert to grayscale
image = Image.open(fileName).convert('L')

store the image dimensions
W, H = image.size[0], image.size[1]

print("input image dims: %d x %d" % (W, H))

compute tile width
w = W/cols

compute tile height based on the aspect ratio and scale o
f the font
h = w/scale

compute number of rows to use in the final grid
rows = int(H/h)

print("cols: %d, rows: %d" % (cols, rows))
print("tile dims: %d x %d" % (w, h))

check if image size is too small
if cols > W or rows > H:
 print("Image too small for specified cols!")
 exit(0)

an ASCII image is a list of character strings
aimg = []

```

```

generate the list of tile dimensions
for j in range(rows):
 y1 = int(j*h)
 y2 = int((j+1)*h)
 # correct the last tile
 if j == rows-1:
 y2 = H
 # append an empty string
 aimg.append("")
 for i in range(cols):
 # crop the image to fit the tile
 x1 = int(i*w)
 x2 = int((i+1)*w)
 # correct the last tile
 if i == cols-1:
 x2 = W

crop the image to extract the tile into another Image object
 img = image.crop((x1, y1, x2, y2))
 # get the average luminance
 avg = int(getAverageL(img))

look up the ASCII character for grayscale value (avg)

```

```

 if moreLevels:
 gsval = gscale1[int((avg*69)/255)]
 else:
 gsval = gscale2[int((avg*9)/255)]
 # append the ASCII character to the string
 aimg[j] += gsval

 # return text image
 return aimg

main() function
def main():
 # create parser

 descStr = "This program converts an image into ASCII art."

 parser = argparse.ArgumentParser(description=descStr)

 # add expected arguments

 parser.add_argument('--
file', dest='imgFile', required=True)

 parser.add_argument('--
scale', dest='scale', required=False)

 parser.add_argument('--
out', dest='outFile', required=False)

 parser.add_argument('--
cols', dest='cols', required=False)

```

```
 parser.add_argument('--
morelevels', dest='moreLevels', action='store_true')

parse arguments

args = parser.parse_args()

imgFile = args.imgFile

set output file

outFile = 'out.txt'

if args.outFile:

 outFile = args.outFile

set scale default as 0.43, which suits a Courier font

scale = 0.43

if args.scale:

 scale = float(args.scale)

set cols

cols = 80

if args.cols:

 cols = int(args.cols)

print('generating ASCII art...')

convert image to ASCII text

aimg = covertImageToAscii(imgFile, cols, scale, args.moreLe
vels)
```

```

open a new text file

f = open(outFile, 'w')

write each string in the list to the new file
for row in aimg:
 f.write(row + '\n')



clean up
f.close()

print("ASCII art written to %s" % outFile)

call main
if __name__ == '__main__':
 main()

```

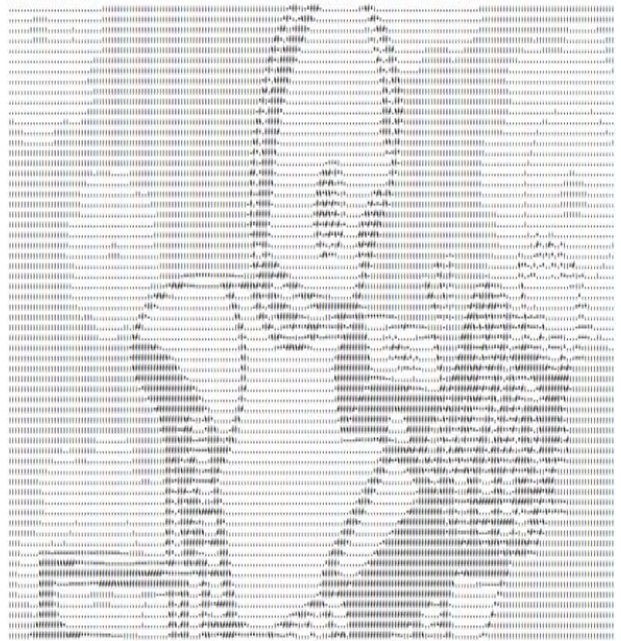
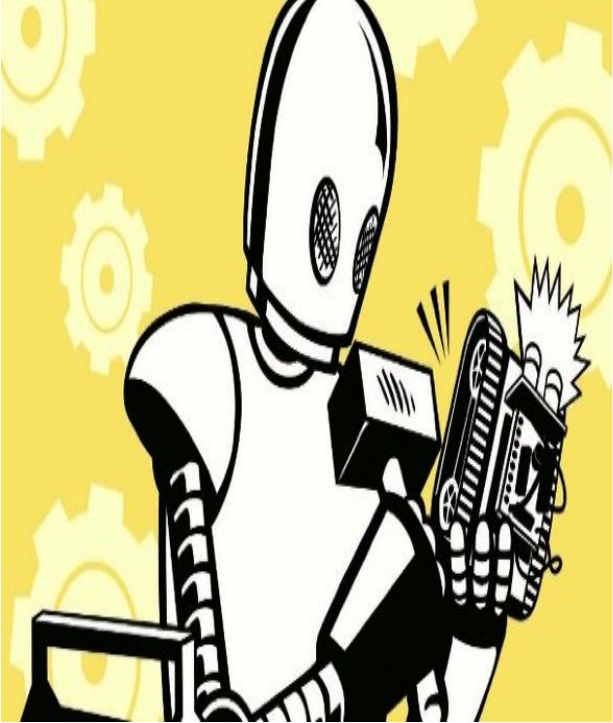
## 6.5 ASCII Art

```
$ python ascii.py --file data/robot.jpg --cols 100
```



## 6-3 ascii.py

ASCII

## 6.6

ASCII

## 6.7

ASCII

1--scale1.0  
scale

□□□□□□□□□□□□□□□□

2□□□□□□□□□□--invert□□□ASCII□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□□□255□□□□□□□□□□

3□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□ASCII□□□□□□□□□□

```
python3 ascii.py --map "@$%^`\"
```

□□□□□□□□□□□6□□□□□□□□□□□ASCII□□□□□  
□“@”□□□□□□0□“.”□□□□□□255□

7□ □□□□□



□□□□□□□□□□□□7-1□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

**7-1**

Python

- Python → PIL →
- → RGB →
- →
- →
- → RGB →

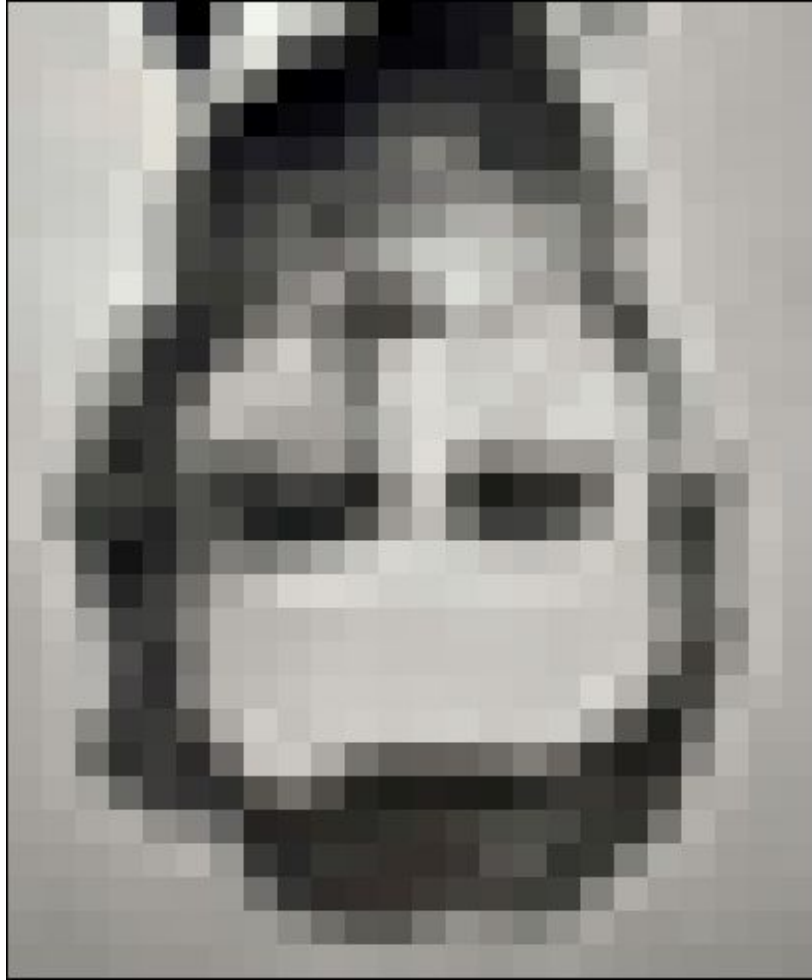


图7-1 笑脸表情

## 7.1 二维数组

二维数组是指具有两个下标的数组。二维数组的每个元素都是一个一维数组。二维数组的声明格式如下：  
数据类型 数组名[M][N];  
其中M表示数组的行数，N表示数组的列数。

1. 二维数组的初始化

2. 二维数组的遍历

7-2  $x$   
y

遍历图像中的每个像素点  $i, j$ ，计算其邻域  $((i+1)*w, (j+1)*h)$  的像素值  $w, h$ ，并计算其平均值  $PIL$ 。

## 7.1.2 平均值

遍历图像中的每个像素点  $i, j$ ，计算其邻域  $((i+1)*w, (j+1)*h)$  的像素值  $w, h$ ，并计算其平均值  $PIL$ 。

$$r, g, b \left( \frac{r_1 + r_2 + \dots + r_N}{N}, \frac{g_1 + g_2 + \dots + g_N}{N}, \frac{b_1 + b_2 + \dots + b_N}{N} \right)$$

遍历图像中的每个像素点  $i, j$ ，计算其邻域  $((i+1)*w, (j+1)*h)$  的像素值  $w, h$ ，并计算其平均值  $PIL$ 。

## 7.1.3 距离

遍历图像中的每个像素点  $i, j$ ，计算其邻域  $((i+1)*w, (j+1)*h)$  的像素值  $w, h$ ，并计算其平均值  $PIL$ 。

遍历图像中的每个像素点  $i, j$ ，计算其邻域  $((i+1)*w, (j+1)*h)$  的像素值  $w, h$ ，并计算其平均值  $PIL$ 。

$$D_{1,2} = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

遍历图像中的每个像素点  $i, j$ ，计算其邻域  $((i+1)*w, (j+1)*h)$  的像素值  $w, h$ ，并计算其平均值  $PIL$ 。

## 7.2 安装

安装 Pillow 和 numpy 库

## 7.3 读取

读取图像文件，返回一个 Image 对象。Image 对象包含图像的元数据和像素数据。RGB 图像有 3 个通道，每个通道是一个字节。7.4 节

### 7.3.1 读取图像

读取图像文件，返回一个 Image 对象。

```
def getImages(imageDir):
 """
 given a directory of images, return a list of Images
 """
 ❶ files = os.listdir(imageDir)
 images = []
 for file in files:
 ❷ filePath = os.path.abspath(os.path.join(imageDir, file))
 try:
```

```
explicit load so we don't run into resource crunch
```

```
③ fp = open(filePath, "rb")

 im = Image.open(fp)

 images.append(im)

 # force loading the image data from file

④ im.load()

 # close the file

⑤ fp.close()

except:

 # skip

 print("Invalid image: %s" % (filePath,))

return images
```

❶ `os.listdir()` `imageDir` 的目录列表  
遍历列表，对每个文件调用 `PIL Image`

❷ `os.path.abspath()` `os.path.join()` 用于  
Python 中路径的表示  
Windows 使用 `\foo\bar`  
Linux 使用 `/c:\foo\bar\`

`PIL Image` 的 `Image.open()` 方法  
Python 中路径的表示



`Image.open()`은 `fp`를 `PIL()`로 변환하여 이미지를  
열어줍니다.

❸ `open()`은 이미지를 `im`로 반환합니다.  
`Image.open()`은 `im`를 반환합니다.

`open()`은 이미지를 `im`로 반환합니다. ❹ `Image.load()`은  
`im`를 `im`로 반환합니다. `im`는 `im`를 반환합니다.  
이미지를 `im`로 반환합니다.

❺ `im`를 `im`로 반환합니다.

## 7.3.2 이미지 처리

이미지를 `im`로 반환합니다. `im`를 `im`로 반환합니다.  
`im`를 `im`로 반환합니다. `im`를 `im`로 반환합니다.

```
def getAverageRGB(image):
```

```
 """
```

```
 return the average color value as (r, g, b) for each input
 image
```

```
 """
```

```
 # get each tile image as a numpy array
```

```
❶ im = np.array(image)
```

```
 # get the shape of each input image
```

```
❷ w,h,d = im.shape
```

```
 # get the average RGB value
```

```
③ return tuple(np.average(im.reshape(w*h, d), axis=0))
```

① numpy Image  
numpy (w, h, d) w h d  
RGB 3 R G B  
② shape RGB  
(w\*h, d) numpy.average()  
③

### 7.3.3

$M \times N$

```
def splitImage(image, size):
```

```
 """
```

```
 given the image and dimensions (rows, cols), return an m*n
 list of images
```

```
 """
```

```
① W, H = image.size[0], image.size[1]
```

```
② m, n = size
```

```
③ w, h = int(W/n), int(H/m)
```

```
 # image list
```

```
 imgs = []
```

```
 # generate a list of dimensions
```

```
 for j in range(m):
```

```

 for i in range(n):
 # append cropped image
 ❷ imgs.append(image.crop((i*w, j*h, (i+1)*w, (j+1)*h)))
 return imgs

```

❶ ❷ ❸  
 ❹

❹  
 image.crop()  
 7.1.1

## 7.3.4

getBestMatchIndex()

```

def getBestMatchIndex(input_avg, avgs):
 """
 return index of the best image match based on average RGB v
 alue distance
 """

 # input image average
 avg = input_avg

```

```
get the closest RGB value to input, based on RGB distance
```

```
index = 0
```

```
❶ min_index = 0
```

```
❷ min_dist = float("inf")
```

```
❸ for val in avgs:
```

```
❹ dist = ((val[0] - avg[0])*(val[0] - avg[0]) +
 (val[1] - avg[1])*(val[1] - avg[1]) +
 (val[2] - avg[2])*(val[2] - avg[2]))
```

```
❺ if dist < min_dist:
```

```
 min_dist = dist
```

```
 min_index = index
```

```
 index += 1
```

```
return min_index
```

avgsRGBinput\_avg  
avgsRGB

RGB❶❷  
0  
❸❹  
❺  
min\_dist  
RGBargsinput\_avg

## 7.3.5 创建图像网格

创建图像网格的函数是 `createImageGrid()`，它接受一个图像列表 `images` 和一个网格尺寸 `M×N`，并返回一个包含所有图像的网格。

```
def createImageGrid(images, dims):
 """
 given a list of images and a grid size (m, n), create a grid of images
 """
 ❶ m, n = dims

 # sanity check
 assert m*n == len(images)

 # get the maximum height and width of the images
 # don't assume they're all equal
 ❷ width = max([img.size[0] for img in images])
 height = max([img.size[1] for img in images])

 # create the target image
 ❸ grid_img = Image.new('RGB', (n*width, m*height))
```

```

paste the tile images into the image grid

for index in range(len(images)):

 ❷ row = int(index/n)

 ❸ col = index - n*row

 ❹ grid_img.paste(images[index], (col*width, row*height))

return grid_img

```

❶ `assert` `createImageGrid()` `assert` `RGB`

❷ `Image` `Image.paste()` `Image.paste()` `Image` `N*row`

```
+ colnames(NB[row, col])
4
5
```

### 7.3.6

```
main
```

```
def createPhotomosaic(target_image, input_images, grid_size
, reuse_images=True):
```

|| || ||

creates a photomosaic given target and input images

|| || ||

```
print('splitting input image...')
```

```
split the target image into tiles
```

```
❶ target_images = splitImage(target_image, grid_size)
```

```
print('finding image matches...')
```

```
for each tile, pick one matching input image
```

```
output_images = []
```

```
for user feedback
```

```
count = 0
```

```
② batch_size = int(len(target_images)/10)
```

```

calculate the average of the input image

avgs = []

for img in input_images:
 ❸ avgs.append(getAverageRGB(img))

for img in target_images:
 # compute the average RGB value of the image
 ❹ avg = getAverageRGB(img)
 # find the matching index of closest RGB value
 # from a list of average RGB values
 ❺ match_index = getBestMatchIndex(avg, avgs)
 ❻ output_images.append(input_images[match_index])
 # user feedback

 ❼
if count > 0 and batch_size > 10 and count % batch_size is
0:
 print('processed %d of %d...' %
(count, len(target_images)))
 count += 1
 # remove the selected image from input if flag set
 ❽ if not reuse_images:
 input_images.remove(match)

print('creating mosaic...')

```



```

create photomosaic image from tiles

⑨ mosaic_image = createImageGrid(output_images, grid_size)

display the mosaic

return mosaic_image

```

`createPhotomosaic()` 函数接收两个参数：要生成光栅图的输入图像列表 **①** 和光栅图的大小 `grid_size`。该函数将输入图像列表中的每个图像转换为 RGB 格式，并计算其平均颜色。然后，它使用 `createImageGrid` 函数将平均颜色图像组合成一个光栅图。最后，它返回生成的光栅图。

**②** `batch_size` 参数指定了每次迭代处理的图像数量。默认值为 **⑦** 100。该参数用于控制内存使用，因为一次加载太多图像可能会导致内存溢出。在代码中，它被用于生成输入图像的批次，并传递给 `createImageGrid` 函数。

**③** 该函数接收两个参数：要生成光栅图的输入图像列表 `images` 和光栅图的大小 `grid_size`。该函数将输入图像列表中的每个图像转换为 RGB 格式，并计算其平均颜色。然后，它使用 `createImageGrid` 函数将平均颜色图像组合成一个光栅图。最后，它返回生成的光栅图。

**⑦** `batch_size` 参数指定了每次迭代处理的图像数量。默认值为 **⑧** 100。该参数用于控制内存使用，因为一次加载太多图像可能会导致内存溢出。在代码中，它被用于生成输入图像的批次，并传递给 `createImageGrid` 函数。

## 7.3.7 参数解析

```
def main():
```

```
 # parse arguments
```

```
 parser = argparse.ArgumentParser(description='Creates a photo mosaic from
```

```
 input images')
```

```
 # add arguments
```

```
 parser.add_argument('--target-image', dest='target_image', required=True)
```

```
 parser.add_argument('--input-folder', dest='input_folder', required=True)
```

```
 parser.add_argument('--grid-size', nargs=2, dest='grid_size', required=True)
```

```
 parser.add_argument('--output-file', dest='outfile', required=False)
```

```
 # Create a 3x3 grid of images
 # Create a 4x4 grid of images
 # Save the mosaic.png
```

## 7.3.8 图像生成

```
 # Load the target image
 # Load the input images
 # Create the mosaic image
```

RGB  
main()

```
print('resizing images...')
```

```
for given grid size, compute the maximum width and height
of tiles
```

```
❶ dims = (int(target_image.size[0]/grid_size[1]),
 int(target_image.size[1]/grid_size[0]))
```

```
print("max tile dims: %s" % (dims,))
```

```
resize
```

```
for img in input_images:
```

```
❷ img.thumbnail(dims)
```

❶ PIL  
Image.thumbnail()

## 7.4

[https://github.com/electronut/pp/tree/master/photomosaic/ photomosaic.py](https://github.com/electronut/pp/tree/master/photomosaic/photomosaic.py)

```
import sys, os, random, argparse
```

```
from PIL import Image
```

```
import imghdr
```

```
import numpy as np
```

```

def getAverageRGB(image):
 """
 return the average color value as (r, g, b) for each input
 image
 """
 # get each tile image as a numpy array
 im = np.array(image)
 # get the shape of each input image
 w,h,d = im.shape
 # get the average RGB value
 return tuple(np.average(im.reshape(w*h, d), axis=0))

```

```

def splitImage(image, size):
 """
 given the image and dimensions (rows, cols), returns an m*n
 list of images
 """
 W, H = image.size[0], image.size[1]
 m, n = size
 w, h = int(W/n), int(H/m)
 # image list
 imgs = []

```

```

 # generate a list of dimensions
 for j in range(m):
 for i in range(n):
 # append cropped image

 imgs.append(image.crop((i*w, j*h, (i+1)*w, (j+1)*h)))

 return imgs

def getImages(imageDir):
 """
 given a directory of images, return a list of Images
 """
 files = os.listdir(imageDir)
 images = []
 for file in files:

 filePath = os.path.abspath(os.path.join(imageDir, file))

 try:

 # explicit load so we don't run into a resource crunch

 fp = open(filePath, "rb")
 im = Image.open(fp)
 images.append(im)

 # force loading image data from file

```

```

 im.load()

 # close the file

 fp.close()

 except:

 # skip

 print("Invalid image: %s" % (filePath,))

 return images

def getImageFileNames(imageDir):
 """
 given a directory of images, return a list of image filenames

 """

 files = os.listdir(imageDir)

 filenames = []

 for file in files:

 filePath = os.path.abspath(os.path.join(imageDir, file))

 try:

 imgType = imghdr.what(filePath)

 if imgType:

 filenames.append(filePath)

 except:

```

```

 # skip

 print("Invalid image: %s" % (filePath,))

 return filenames

def getBestMatchIndex(input_avg, avgs):
 """
 return index of the best image match based on average RGB v
 alue distance

 """

 # input image average

 avg = input_avg

 # get the closest RGB value to input, based on RGB distance

 index = 0

 min_index = 0

 min_dist = float("inf")

 for val in avgs:

 dist = ((val[0] - avg[0])*(val[0] - avg[0]) +
 (val[1] - avg[1])*(val[1] - avg[1]) +
 (val[2] - avg[2])*(val[2] - avg[2]))

 if dist < min_dist:

 min_dist = dist

```

```
 min_index = index
 index += 1

 return min_index
```

```
def createImageGrid(images, dims):
```

```
 """
```

```
 given a list of images and a grid size (m, n), create a grid of images
```

```
 """
```

```
 m, n = dims
```

```
 # sanity check
```

```
 assert m*n == len(images)
```

```
 # get the maximum height and width of the images
```

```
 # don't assume they're all equal
```

```
 width = max([img.size[0] for img in images])
```

```
 height = max([img.size[1] for img in images])
```

```
 # create the target image
```

```
 grid_img = Image.new('RGB', (n*width, m*height))
```



```

 # paste the tile images into the image grid
 for index in range(len(images)):
 row = int(index/n)
 col = index - n*row

 grid_img.paste(images[index], (col*width, row*height))

 return grid_img

def createPhotomosaic(target_image, input_images, grid_size
, reuse_images=True):
 """
 creates photomosaic given target and input images
 """

 print('splitting input image...')
 # split the target image into tiles
 target_images = splitImage(target_image, grid_size)

 print('finding image matches...')
 # for each tile, pick one matching input image

```

```

output_images = []

for user feedback

count = 0

batch_size = int(len(target_images)/10)

calculate the average of the input image

avgs = []

for img in input_images:
 avgs.append(getAverageRGB(img))

for img in target_images:
 # compute the average RGB value of the image
 avg = getAverageRGB(img)
 # find the matching index of closest RGB value
 # from a list of average RGB values
 match_index = getBestMatchIndex(avg, avgs)
 output_images.append(input_images[match_index])
 # user feedback

if count > 0 and batch_size > 10 and count % batch_size is
0:
 print('processed %d of %d...' %
(count, len(target_images)))
 count += 1

```

```

 # remove the selected image from input if flag set
 if not reuse_images:
 input_images.remove(match)

 print('creating mosaic...')

 # create photomosaic image from tiles
 mosaic_image = createImageGrid(output_images, grid_size)

 # display the mosaic
 return mosaic_image

gather our code in a main() function
def main():

 # command line arguments are in sys.argv[1], sys.argv[2], .
 ..

 # sys.argv[0] is the script name itself and can be ignored

 # parse arguments

 parser = argparse.ArgumentParser(description='Creates a photomosaic from

 input images')

 # add arguments

 parser.add_argument('--target-image', dest='target_image', required=True)

```

```

 parser.add_argument('--input-
folder', dest='input_folder', required=True)

 parser.add_argument('--grid-
size', nargs=2, dest='grid_size', required=True)

 parser.add_argument('--output-
file', dest='outfile', required=False)

args = parser.parse_args()

INPUTS

target image
target_image = Image.open(args.target_image)

input images
print('reading input folder...')
input_images = getImages(args.input_folder)

check if any valid input images found
if input_images == []:

print('No input images found in %s. Exiting.' % (args.input
_folder,))

 exit()

```

```
shuffle list to get a more varied output?
random.shuffle(input_images)

size of the grid

grid_size = (int(args.grid_size[0]), int(args.grid_size[1])
)

output
output_filename = 'mosaic.png'
if args.outfile:
 output_filename = args.outfile

reuse any image in input
reuse_images = True

resize the input to fit the original image size?
resize_input = True

END INPUTS

print('starting photomosaic creation...')
```

[illegible]

```

mosaic_image.save(output_filename, 'PNG')

print("saved output to %s" % (output_filename,))

print('done.')

standard boilerplate to call the main() function
to begin the program
if __name__ == '__main__':
 main()

```

## 7.5 实验

实验内容

```
$ python photomosaic.py --target-image test-
data/cherai.jpg --input-folder
```

```
test-data/set6/ --grid-size 128 128
```

```
reading input folder...
```

```
starting photomosaic creation...
```

```
resizing images...
```

```
max tile dims: (23, 15)
```

```
splitting input image...
```

```
finding image matches...
```

```
processed 1638 of 16384 ...
```

```
processed 3276 of 16384 ...
```

processed 4914 of 16384 ...

creating mosaic...

saved output to mosaic.png

done.

7-3 a b c



(a)



(b)



(c)

7-3

## 7.6

## 7.7



[illegible]

**10000000000000000007-10**

[illegible]

# createImageGrid()

### 3. `getBestMatchIndex()`

SciPy K-D

`scipy.spatial.KDTree` K-D

SciPy K-D

<http://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html>



**8-1**

https://github.com/electronut/pp/images/□  
□□□□□□□□□□□□□□□□

Python

- ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
- ☐ ☐ ☐ ☐
- ☐ Pillow ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
- ☐ Pillow ☐ ☐ ☐ ☐ ☐

[illegible]

Practice” [1]\_



□8-1 □□□□□□□□□□□□□□□□ [2]

## 8.1 ☐☐☐☐

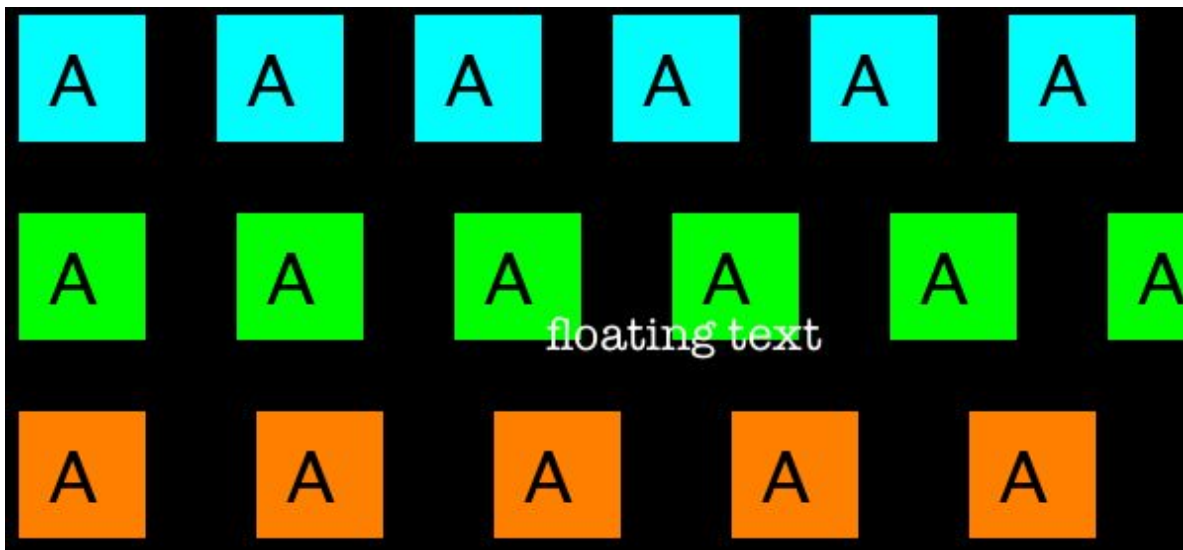
[illegible]

### 8.1.1 □□□□□□□□□□

[illegible]

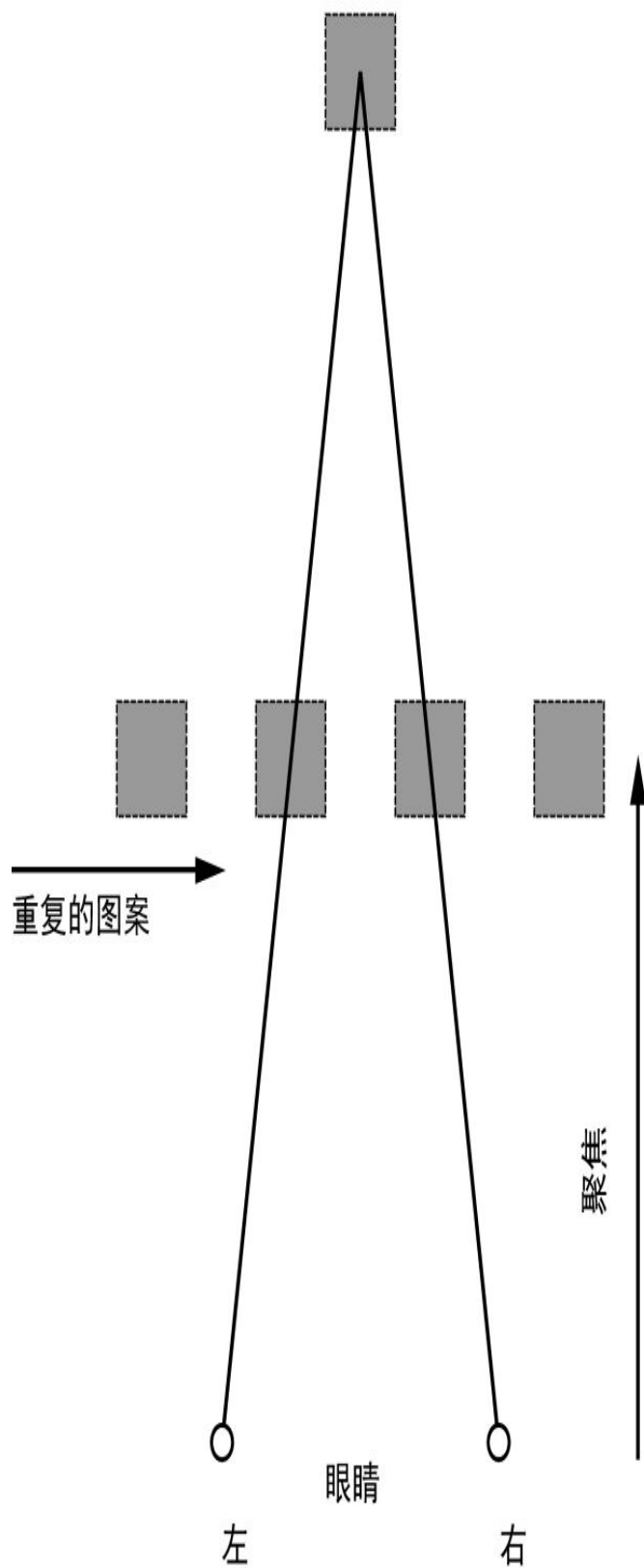
8-2 3 A A

`<float>` “floating text”

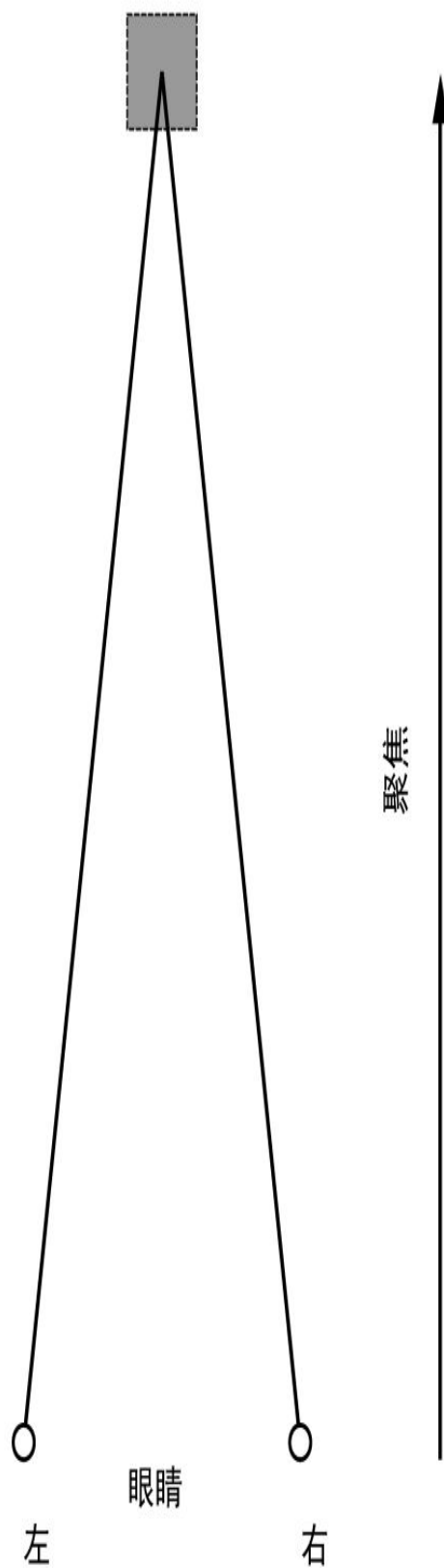
[illegible]

□8-2    □□□□□□□□

汇聚：感知到的立体物体



汇聚：真实距离的物体



### 图8-3 鳐鱼的外形特征

鳐鱼的外形特征与图8-2所示的鲨鱼相似，但其身体扁平，胸鳍扩大成翼状，尾鳍呈深叉形。鳐鱼的身体扁平，胸鳍扩大成翼状，尾鳍呈深叉形。

#### 8.1.2 鳐鱼

“鳐鱼”是一种软骨鱼类，其身体扁平，胸鳍扩大成翼状，尾鳍呈深叉形。鳐鱼的身体扁平，胸鳍扩大成翼状，尾鳍呈深叉形。

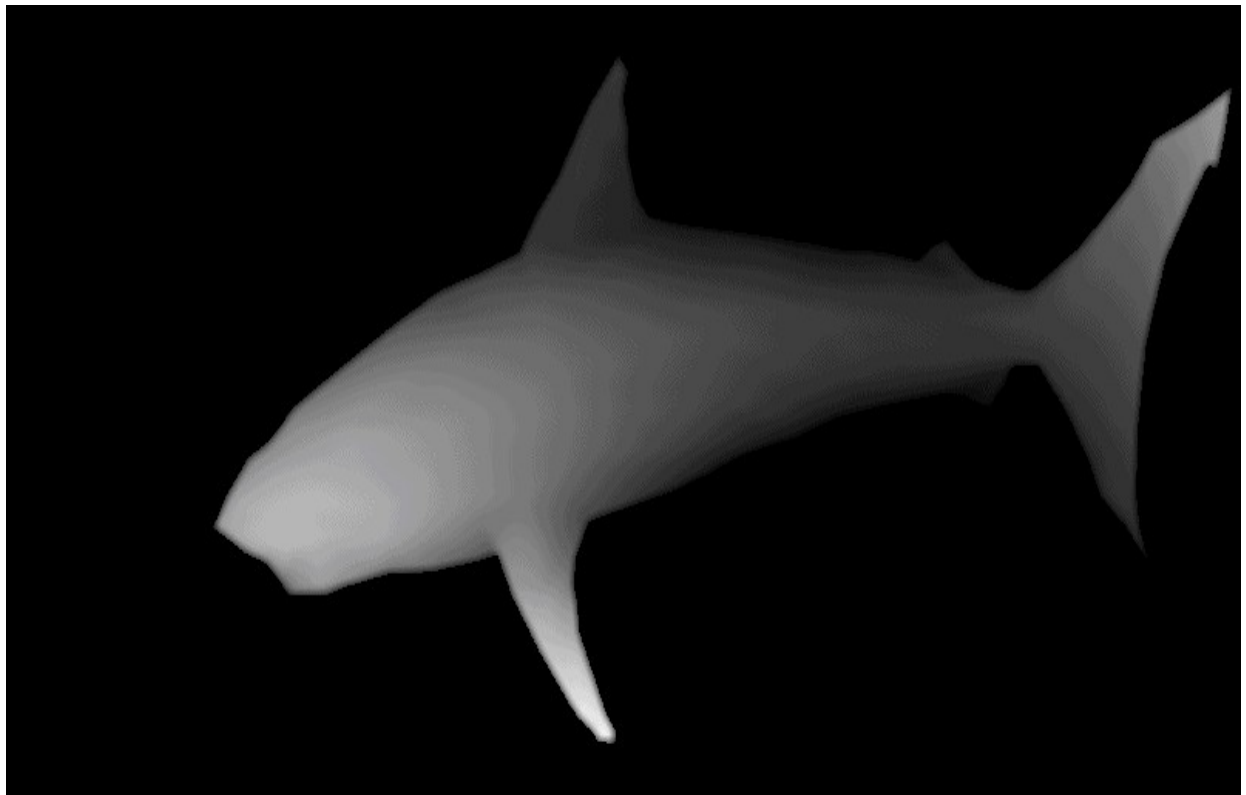


图8-4 鳐鱼

[illegible][illegible]

`uint8_t array[256];`

[illegible]

**1**

2□□□□□□□□□□□□□□“□□□”□□□□□

3

4

5 □ □ □ □ □ □ □ □ □ □ □ □ □

## 8.2 ☐☐☐☐

Pillow

## 8.3 图像

图像是计算机中存储和处理的数据。图像数据通常以像素的形式表示，每个像素由红、绿、蓝三个颜色通道组成。在计算机中，图像通常以二维数组的形式存储，每个元素代表一个像素的颜色值。本章将介绍如何使用Python中的OpenCV库来处理图像，包括图像的读取、显示、裁剪、旋转、缩放等操作。本章还将介绍如何使用OpenCV库中的高级功能，如特征检测、图像识别等。

### 8.4 图像

### 8.3.1 创建图像

使用`cv2.createTiledImage()`函数可以创建一个由指定图像组成的新图像。该函数的原型如下：

```
def createTiledImage(tile, dims, width, height)
```

# tile a graphics file to create an intermediate image of a set size

```
def createTiledImage(tile, dims):
```

```
 # create the new image
```

```
 ❶ img = Image.new('RGB', dims)
```

```
 W, H = dims
```

```
 w, h = tile.size
```

```
 # calculate the number of tiles needed
```

```
 ❷ cols = int(W/w) + 1
```

```
 ❸ rows = int(H/h) + 1
```

```
 # paste the tiles into the image
```

```
 for i in range(rows):
```

```
 for j in range(cols):
```



```

④ img.paste(tile, (j*w, i*h))

output the image

return img

```

❶ `dims` Python `PIL` `Image` `dims` `width`, `height` ❷ `Image` ❸ `ImageDraw` ❹ `Image.paste(tile, (j*w, i*h))`

## 8.3.2 创建随机瓦片

`createRandomTile()`

```

create an image tile filled with random circles

def createRandomTile(dims):

 # create image

 ❶ img = Image.new('RGB', dims)

 ❷ draw = ImageDraw.Draw(img)

 # set the radius of a random circle to 1% of

 # width or height, whichever is smaller

 ❸ r = int(min(*dims)/100)

```

```

 # number of circles

❹ n = 1000

 # draw random circles

 for i in range(n):

 # -
 r makes sure that the circles stay inside and aren't cut of
 f

at the edges of the image so that they'll look better whe
n tiled

❺ x, y = random.randint(0, dims[0]-
r), random.randint(0, dims[1]-r)

❻
fill = (random.randint(0, 255), random.randint(0, 255),
 random.randint(0, 255))

❼ draw.ellipse((x-r, y-r, x+r, y+r), fill)

 return img

```

❶ `dim` is a tuple of the image dimensions. `ImageDraw.Draw()` is a method of the `ImageDraw` module. `1/100` is a float value. `Python` is a string. `dim` is a tuple. `min()` is a function.

❷ `random.randint()` is a function that returns a random integer. `[0, width-r]` is a list. `[0, height-r]` is a list. `x` and `y` are variables. `“-r”` is a string. `width` and `height` are variables. `-r` is a variable.

이 코드는 256x256 픽셀의 이미지를 생성하고, 각 픽셀의 값을 0~255 범위의 랜덤한 값으로 설정합니다.

이 코드는 256x256 픽셀의 이미지를 생성하고, 각 픽셀의 값을 0~255 범위의 랜덤한 값으로 설정합니다. ⑥ `randint(0, 255)`는 0~255 범위의 랜덤한 값을 생성합니다. ⑦ `draw.ellipse()`는 주어진 좌표와 반경으로 타원을 그립니다. `(x-r, y-r)`와 `(x+r, y+r)`는 타원의 좌상단과 우하단 좌표입니다. `r`은 반경입니다.

이 코드는 Python에서 이미지를 생성하고 저장하는 방법을 보여줍니다.

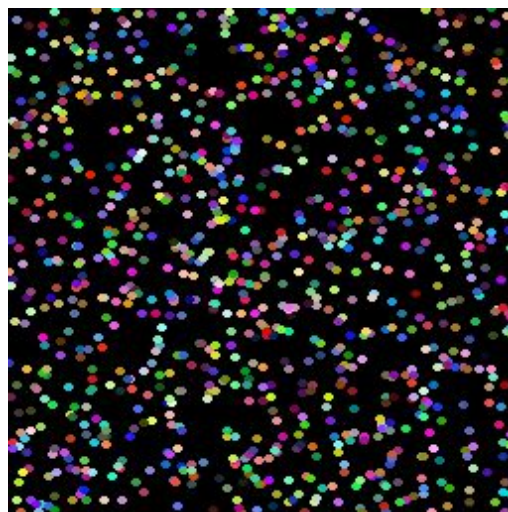
```
>>> import autos

>>> img = autos.createRandomTile((256, 256))

>>> img.save('out.png')

>>> exit()
```

8-5 이미지 생성하기



8-5 `createRandomTile()` 함수를 사용한 이미지 생성

Figure 8-5: A diagram showing a 3D scene with a camera and a depth map. The scene contains a table and a chair. The camera is positioned to the left, and the depth map is shown as a grid of values representing the distance of each point in the scene from the camera.

### 8.3.3 createAutostereogram()

The function `createAutostereogram()` is defined in the `createAutostereogram.py` file.

```
def createAutostereogram(dmap, tile):
 # convert the depth map to a single channel if needed
 ❶ if dmap.mode is not 'L':
 dmap = dmap.convert('L')

 # if no image is specified for a tile, create a random circles tile
 ❷ if not tile:
 tile = createRandomTile((100, 100))

 # create an image by tiling
 ❸ img = createTiledImage(tile, dmap.size)

 # create a shifted image using depth map values
 ❹ sImg = img.copy()

 # get access to image pixels by loading the Image object first
 ❺ pixD = dmap.load()
 pixS = sImg.load()

 # shift pixels horizontally based on depth map
```

```

1 for i in range(10):
2 for j in range(10):
3 img[i, j] = 0
4 for k in range(10):
5 img[i, j, k] = 0
6 img[i, j, k] = 0
7 img[i, j, k] = 0
8 img[i, j, k] = 0
9 img[i, j, k] = 0
10 img[i, j, k] = 0
11 img[i, j, k] = 0
12 img[i, j, k] = 0
13 img[i, j, k] = 0
14 img[i, j, k] = 0
15 img[i, j, k] = 0
16 img[i, j, k] = 0
17 img[i, j, k] = 0
18 img[i, j, k] = 0
19 img[i, j, k] = 0
20 img[i, j, k] = 0
21 img[i, j, k] = 0
22 img[i, j, k] = 0
23 img[i, j, k] = 0
24 img[i, j, k] = 0
25 img[i, j, k] = 0
26 img[i, j, k] = 0
27 img[i, j, k] = 0
28 img[i, j, k] = 0
29 img[i, j, k] = 0
30 img[i, j, k] = 0
31 img[i, j, k] = 0
32 img[i, j, k] = 0
33 img[i, j, k] = 0
34 img[i, j, k] = 0
35 img[i, j, k] = 0
36 img[i, j, k] = 0
37 img[i, j, k] = 0
38 img[i, j, k] = 0
39 img[i, j, k] = 0
40 img[i, j, k] = 0
41 img[i, j, k] = 0
42 img[i, j, k] = 0
43 img[i, j, k] = 0
44 img[i, j, k] = 0
45 img[i, j, k] = 0
46 img[i, j, k] = 0
47 img[i, j, k] = 0
48 img[i, j, k] = 0
49 img[i, j, k] = 0
50 img[i, j, k] = 0
51 img[i, j, k] = 0
52 img[i, j, k] = 0
53 img[i, j, k] = 0
54 img[i, j, k] = 0
55 img[i, j, k] = 0
56 img[i, j, k] = 0
57 img[i, j, k] = 0
58 img[i, j, k] = 0
59 img[i, j, k] = 0
60 img[i, j, k] = 0
61 img[i, j, k] = 0
62 img[i, j, k] = 0
63 img[i, j, k] = 0
64 img[i, j, k] = 0
65 img[i, j, k] = 0
66 img[i, j, k] = 0
67 img[i, j, k] = 0
68 img[i, j, k] = 0
69 img[i, j, k] = 0
70 img[i, j, k] = 0
71 img[i, j, k] = 0
72 img[i, j, k] = 0
73 img[i, j, k] = 0
74 img[i, j, k] = 0
75 img[i, j, k] = 0
76 img[i, j, k] = 0
77 img[i, j, k] = 0
78 img[i, j, k] = 0
79 img[i, j, k] = 0
80 img[i, j, k] = 0
81 img[i, j, k] = 0
82 img[i, j, k] = 0
83 img[i, j, k] = 0
84 img[i, j, k] = 0
85 img[i, j, k] = 0
86 img[i, j, k] = 0
87 img[i, j, k] = 0
88 img[i, j, k] = 0
89 img[i, j, k] = 0
90 img[i, j, k] = 0
91 img[i, j, k] = 0
92 img[i, j, k] = 0
93 img[i, j, k] = 0
94 img[i, j, k] = 0
95 img[i, j, k] = 0
96 img[i, j, k] = 0
97 img[i, j, k] = 0
98 img[i, j, k] = 0
99 img[i, j, k] = 0
100 img[i, j, k] = 0

```

⑧  $a_i = a_i + w a_i x_i /$   
 $y$

$\Delta_i b_i = b_{i-w} + \delta_t$

$b_i$

⑧

⑩ ⑨

## 8.3.4

main()

# create a parser

```
parser = argparse.ArgumentParser(description="Autostereogram
s...")
```

# add expected arguments

```
① parser.add_argument('--
depth', dest='dmFile', required=True)
```

```
parser.add_argument('--
tile', dest='tileFile', required=False)
```

```
parser.add_argument('--
out', dest='outFile', required=False)
```

```

parse args

args = parser.parse_args()

set the output file

outFile = 'as.png'

if args.outFile:

 outFile = args.outFile

set tile

tileFile = False

if args.tileFile:

 tileFile = Image.open(args.tileFile)

```

❶ `argparse` `as.png`

## 8.4

<https://github.com/electronut/pp/blob/master/autos/autos.py>

```

import sys, random, argparse

from PIL import Image, ImageDraw

create spacing/depth example

```

```

def createSpacingDepthExample():

 tiles = [Image.open('test/a.png'), Image.open('test/b.png')
 ,

 Image.open('test/c.png')]

 img = Image.new('RGB', (600, 400), (0, 0, 0))

 spacing = [10, 20, 40]

 for j, tile in enumerate(tiles):
 for i in range(8):
 img.paste(tile, (10 + i*(100 + j*10), 10 + j*100))

 img.save('sdepth.png')

create an image filled with random circles
def createRandomTile(dims):

 # create image

 img = Image.new('RGB', dims)

 draw = ImageDraw.Draw(img)

 # set the radius of a random circle to 1% of

 # width or height, whichever is smaller

 r = int(min(*dims)/100)

 # number of circles

 n = 1000

 # draw random circles

 for i in range(n):

```



```

 # -
r makes sure that the circles stay inside and aren't cut of
f

at the edges of the image so that they'll look better whe
n tiled

 x, y = random.randint(0, dims[0]-
r), random.randint(0, dims[1]-r)

fill = (random.randint(0, 255), random.randint(0, 255),
 random.randint(0, 255))

 draw.ellipse((x-r, y-r, x+r, y+r), fill)

return image

return img

tile a graphics file to create an intermediate image of a
set size

def createTiledImage(tile, dims):

 # create the new image

 img = Image.new('RGB', dims)

 W, H = dims

 w, h = tile.size

 # calculate the number of tiles needed

 cols = int(W/w) + 1

 rows = int(H/h) + 1

 # paste the tiles into the image

```

```

 for i in range(rows):
 for j in range(cols):
 img.paste(tile, (j*w, i*h))
output the image
return img

create a depth map for testing
def createDepthMap(dims):
 dmap = Image.new('L', dims)
 dmap.paste(10, (200, 25, 300, 125))
 dmap.paste(30, (200, 150, 300, 250))
 dmap.paste(20, (200, 275, 300, 375))
 return dmap

given a depth map image and an input image,
create a new image with pixels shifted according to depth
def createDepthShiftedImage(dmap, img):
 # size check
 assert dmap.size == img.size

create shifted image
sImg = img.copy()

```

```

get pixel access
pixD = dmap.load()
pixS = sImg.load()

shift pixels output based on depth map
cols, rows = sImg.size
for j in range(rows):
 for i in range(cols):
 xshift = pixD[i, j]/10
 xpos = i - 140 + xshift
 if xpos > 0 and xpos < cols:
 pixS[i, j] = pixS[xpos, j]

return shifted image
return sImg

given a depth map (image) and an input image,
create a new image with pixels shifted according to depth
def createAutostereogram(dmap, tile):
 # convert the depth map to a single channel if needed
 if dmap.mode is not 'L':
 dmap = dmap.convert('L')

if no image is specified for a tile, create a random circles tile
if not tile:

```

```

 tile = createRandomTile((100, 100))

create an image by tiling

img = createTiledImage(tile, dmap.size)

create a shifted image using depth map values

sImg = img.copy()

get access to image pixels by loading the Image object first

pixD = dmap.load()

pixS = sImg.load()

shift pixels horizontally based on depth map

cols, rows = sImg.size

for j in range(rows):
 for i in range(cols):
 xshift = pixD[i, j]/10
 xpos = i - tile.size[0] + xshift
 if xpos > 0 and xpos < cols:
 pixS[i, j] = pixS[xpos, j]

return shifted image

return sImg

main() function

def main():

```

```
use sys.argv if needed

print('creating autostereogram...')

create parser

parser = argparse.ArgumentParser(description="Autostereogram
s...")

add expected arguments

 parser.add_argument('--
depth', dest='dmFile', required=True)

 parser.add_argument('--
tile', dest='tileFile', required=False)

 parser.add_argument('--
out', dest='outFile', required=False)

parse args

args = parser.parse_args()

set the output file

outFile = 'as.png'

if args.outFile:

 outFile = args.outFile

set tile

tileFile = False

if args.tileFile:

 tileFile = Image.open(args.tileFile)

open depth map

dmImg = Image.open(args.dmFile)
```

```

create stereogram

asImg = createAutostereogram(dmImg, tileFile)

write output

asImg.save(outFile)

call main

if __name__ == '__main__':
 main()

```

## 8.5 深度图生成

我们使用 `stool-depth.png` 作为输入图像

```
$ python3 autos.py --depth data/stool-depth.png
```

8-6 深度图生成结果

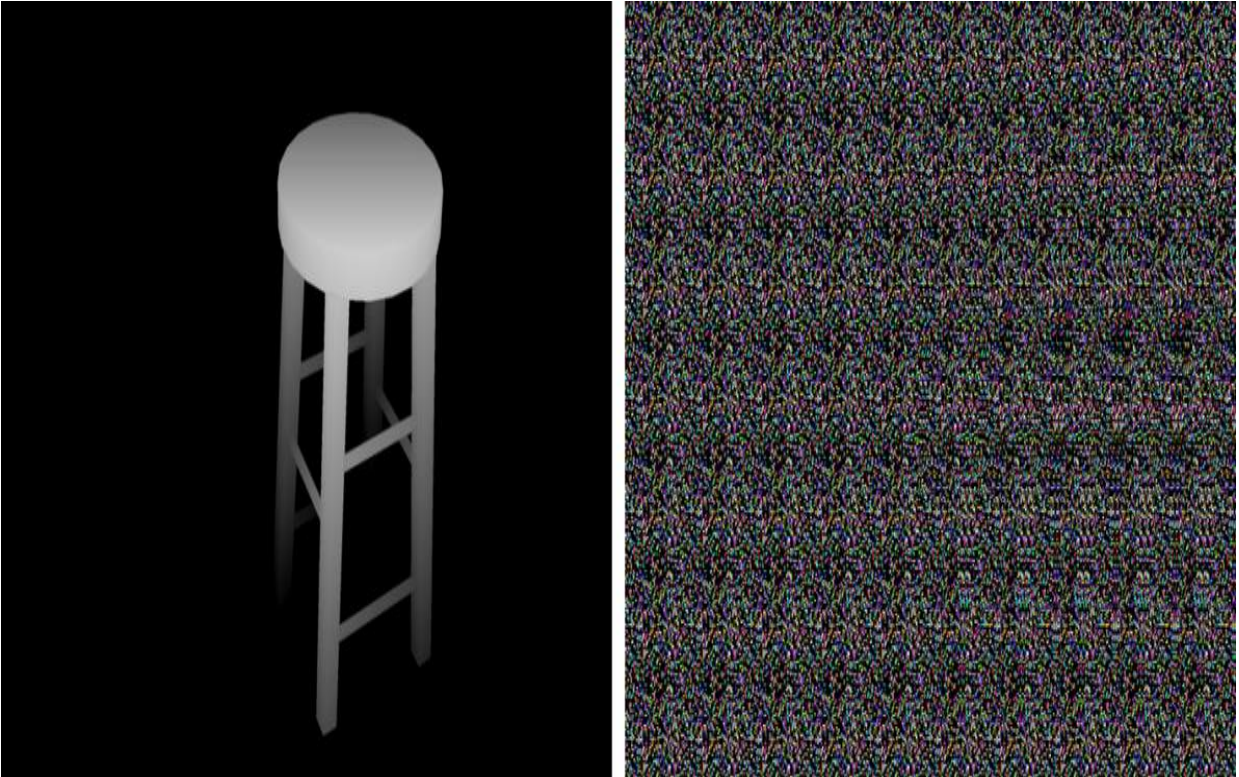


图8-6 autos.py的输入

深度图数据文件 stool-  
depth.png 和 纹理图文件 escher-tile.jpg [\[3\]](#)  
运行命令

```
$ python3 autos.py --depth data/stool-depth.png --
tile data/escher-tile.jpg
```

图8-7 运行结果

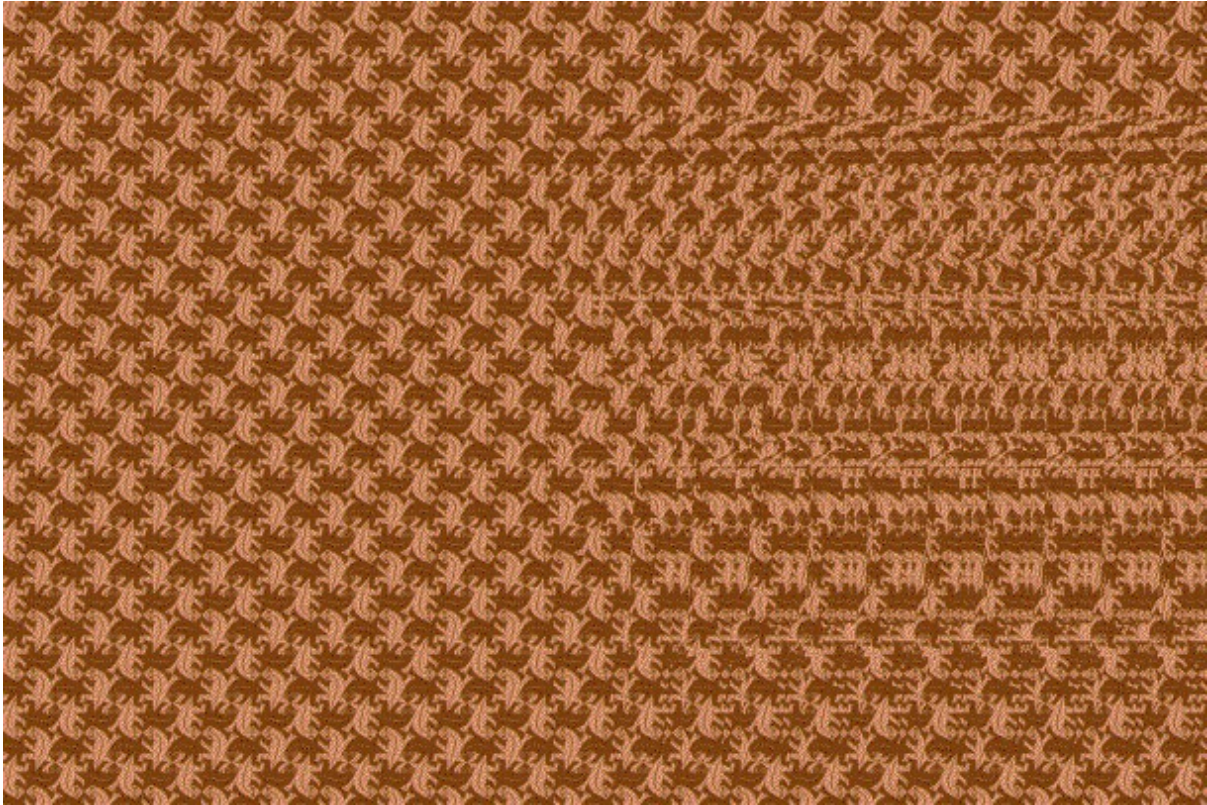


图8-7 在代码文件autos.py中

## 8.6 图

图8-6展示了在代码文件autos.py中，`Image.open()`方法用于打开图像文件。该方法的第一个参数是图像文件的路径，第二个参数是图像的模式。在图8-6中，图像的模式被设置为`'RGB'`。

## 8.7 图

图8-7展示了在代码文件autos.py中，`Image.paste()`方法用于将图像的一部分粘贴到另一图像中。该方法的第一个参数是源图像，第二个参数是目标图像，第三个参数是粘贴的坐标。

1. 在代码文件autos.py中，`Image.paste()`方法用于将图像的一部分粘贴到另一图像中。该方法的第一个参数是源图像，第二个参数是目标图像，第三个参数是粘贴的坐标。  
在代码文件autos.py中，`Image.paste()`方法用于将图像的一部分粘贴到另一图像中。该方法的第一个参数是源图像，第二个参数是目标图像，第三个参数是粘贴的坐标。



2 用 10 分钟时间，用 SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。

3 用 SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。  
<http://sketchup.com/>  
SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。  
<https://www.youtube.com/watch?v=fDzNJYi6Bok/>

---

[1] [http://colorstereo.com/texts\\_.txt/practice.htm](http://colorstereo.com/texts_.txt/practice.htm)

[2] 用 10 分钟时间，用 SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。

[3] <http://calculus-geometry.hubpages.com/hub/Free-M-C-Escher-Tessellation-Background-Patterns-Tiling-Lizard-Background/>

用 10 分钟时间，用 SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。

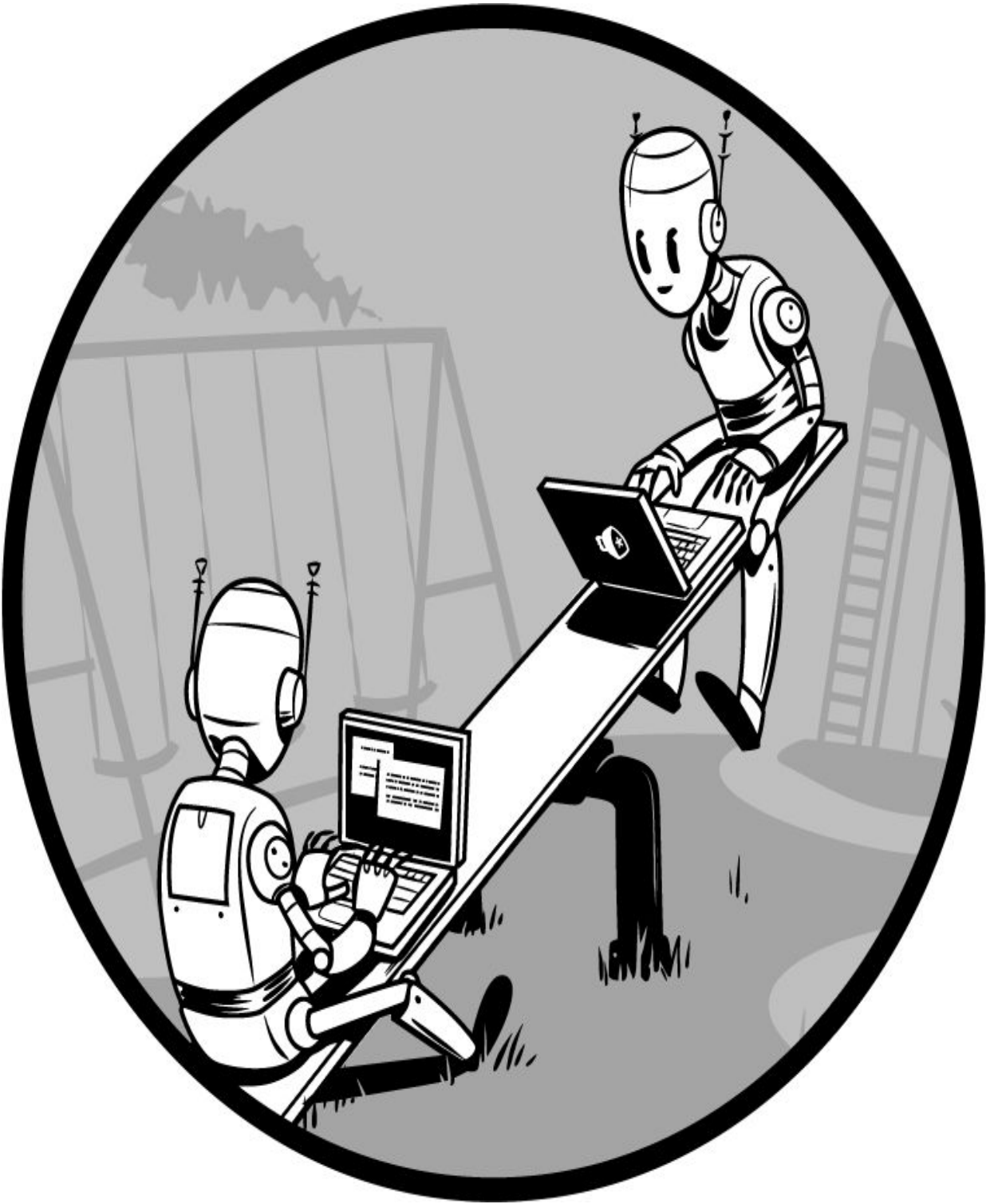
“用 10 分钟时间，用 SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。”

用 10 分钟时间，用 SketchUp 软件，制作一个 10 分钟的视频，展示你的作品。

[illegible]

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ "

——Edwin A. Abbott, *Flatland: A Romance of Many Dimensions*



## 第9章 OpenGL



本章主要介绍OpenGL和GLFW库的使用。OpenGL是一个跨平台的GPU渲染API，GLFW是一个跨平台的窗口和输入库。本章将介绍如何使用OpenGL和GLFW来创建一个简单的窗口，并在其中渲染一个3D模型。本章还将介绍如何使用GLSL来编写着色器程序。本章最后将介绍如何使用Python来调用OpenGL和GLFW库。

9-1 本章结构

GPU是一个专门用于处理图形数据的处理器。它通常与CPU一起使用，以加速图形渲染。本章将介绍如何使用GPU来渲染3D模型。本章还将介绍如何使用GLSL来编写着色器程序。本章最后将介绍如何使用Python来调用OpenGL和GLFW库。

Python是一种高级编程语言，它提供了丰富的库和工具。本章将介绍如何使用Python来调用OpenGL和GLFW库。本章还将介绍如何使用Python来编写一个简单的3D渲染程序。本章最后将介绍如何使用Python来调用OpenGL和GLFW库。

第11章 PyOpenGL 与 OpenGL 与 Python 的交互  
本章主要介绍

OpenGL 的“图形”库，它提供了对 OpenGL 的接口。OpenGL 是一个跨平台的图形库，它提供了对 OpenGL 的接口。OpenGL 是一个跨平台的图形库，它提供了对 OpenGL 的接口。OpenGL 是一个跨平台的图形库，它提供了对 OpenGL 的接口。

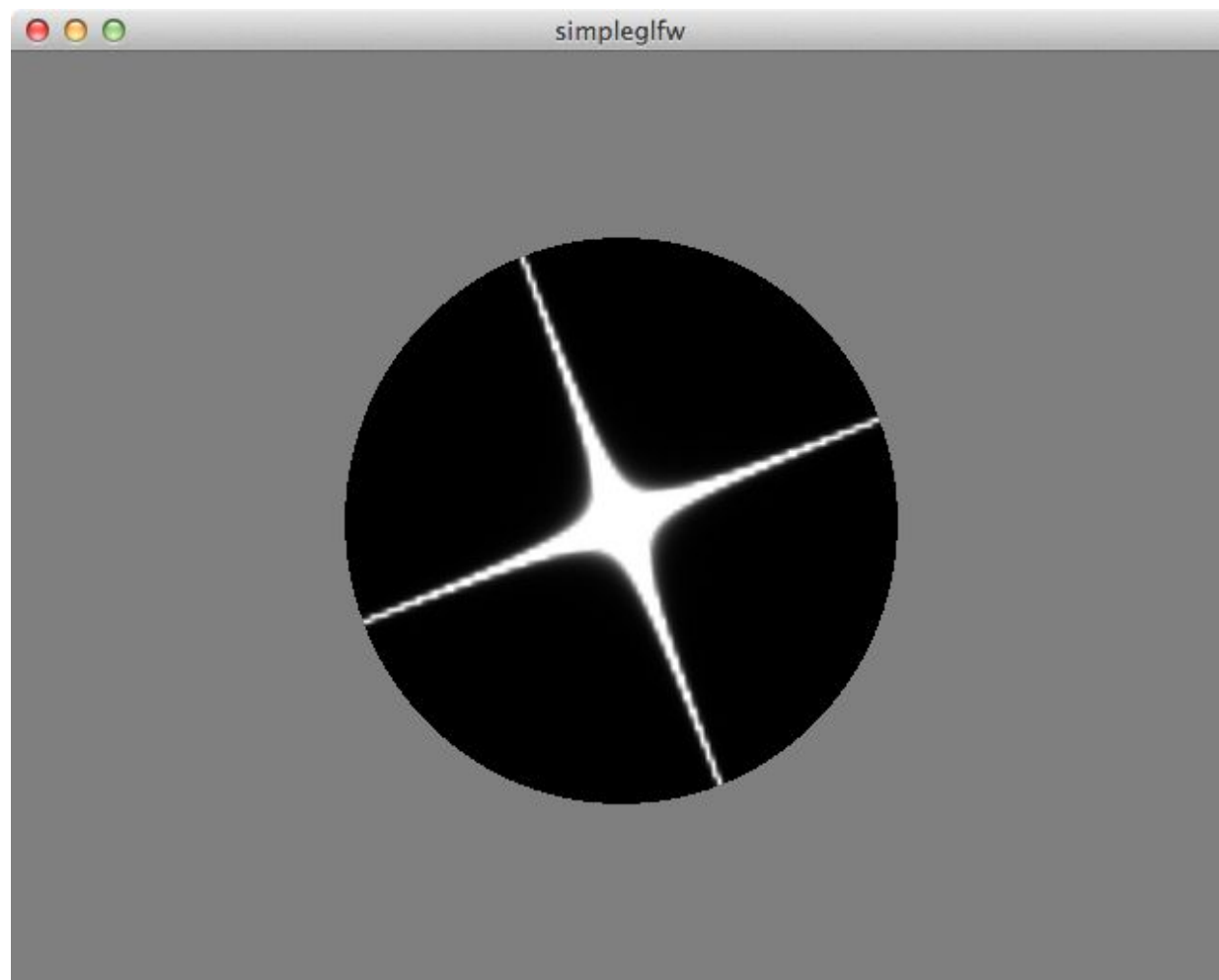


图9-1 一个简单的OpenGL窗口

[illegible]

- OpenGL GLFW
- GLSL
- 
- 

## OpenGLのインストール

## 9.1 OpenGL

OpenGL is a cross-platform API for rendering 2D and 3D graphics. It is a standard for writing applications that use computer graphics hardware to render 2D and 3D graphics. OpenGL is a cross-platform API for rendering 2D and 3D graphics. It is a standard for writing applications that use computer graphics hardware to render 2D and 3D graphics.

OpenGL

```
import sys

from OpenGL.GLUT import *

from OpenGL.GL import *

def display():

 glClear (GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)

 glColor3f (1.0, 1.0, 0.0)

 glBegin(GL_QUADS)
```

```
glVertex3f (-0.5, -0.5, 0.0)
glVertex3f (0.5, -0.5, 0.0)
glVertex3f (0.5, 0.5, 0.0)
glVertex3f (-0.5, 0.5, 0.0)
glEnd()
glFlush();
```

```
glutInit(sys.argv)
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)
glutInitWindowSize(400, 400)
glutCreateWindow("oldgl")
glutDisplayFunc(display)
glutMainLoop()
```

□9-2□□□□□□

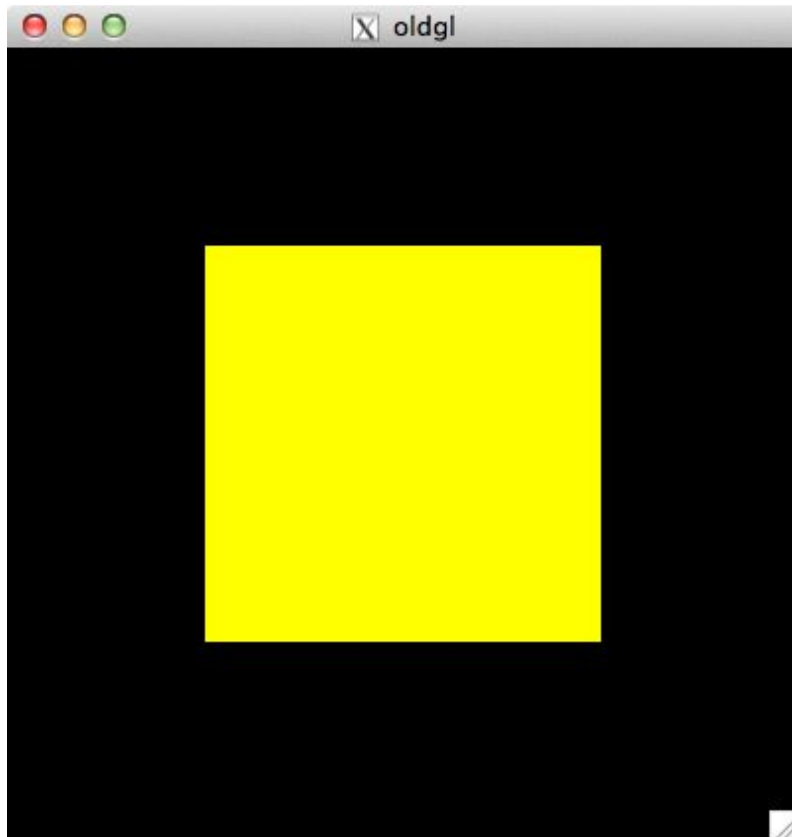


図9-2 OpenGLの起動画面

OpenGLは、GL\_QUADという primitive shape を GPU に渡して描画する。この形状は、2D の図形であり、3D の図形ではない。この形状は、2D の図形であり、3D の図形ではない。

## 9.2 OpenGLの起動画面

OpenGLの起動画面は、図9-3に示すように、黒い背景に黄色い正方形が表示される。この画面は、OpenGLの起動時に表示される。この画面は、OpenGLの起動時に表示される。



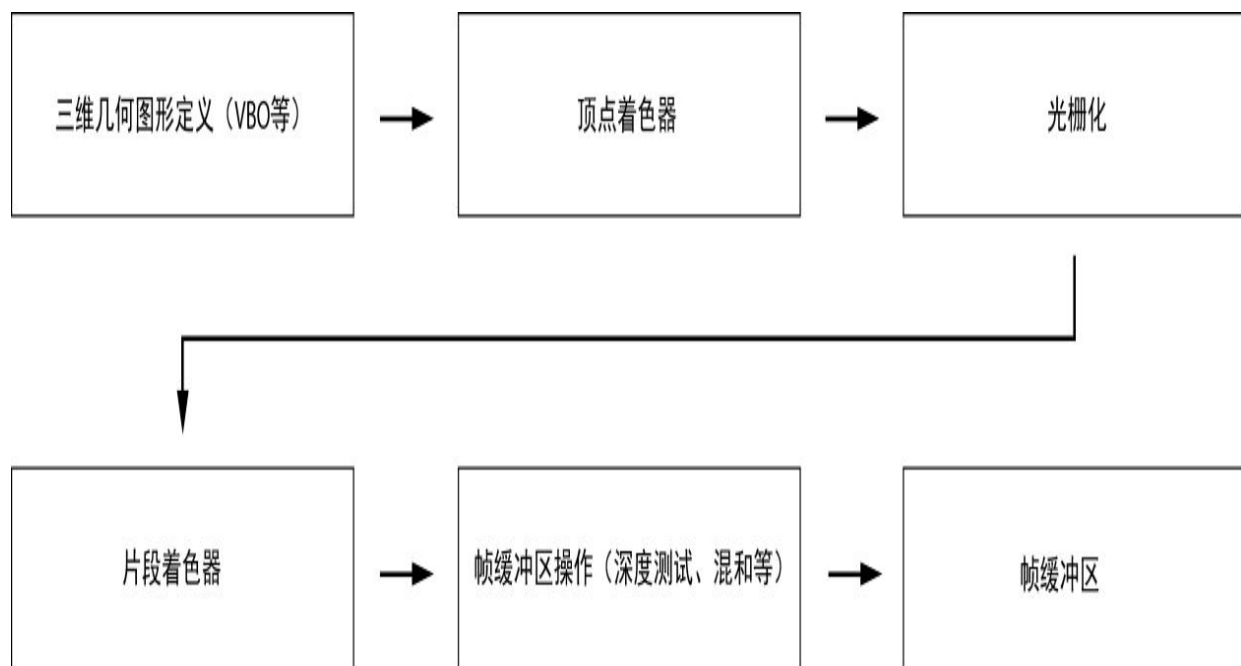


图9-3 OpenGL渲染流程

OpenGL渲染流程如下：  
 1. 定义几何体（VBO等）  
 2. 顶点着色器  
 3. 光栅化  
 4. 片段着色器  
 5. 帧缓冲区操作（深度测试、混和等）  
 6. 帧缓冲区

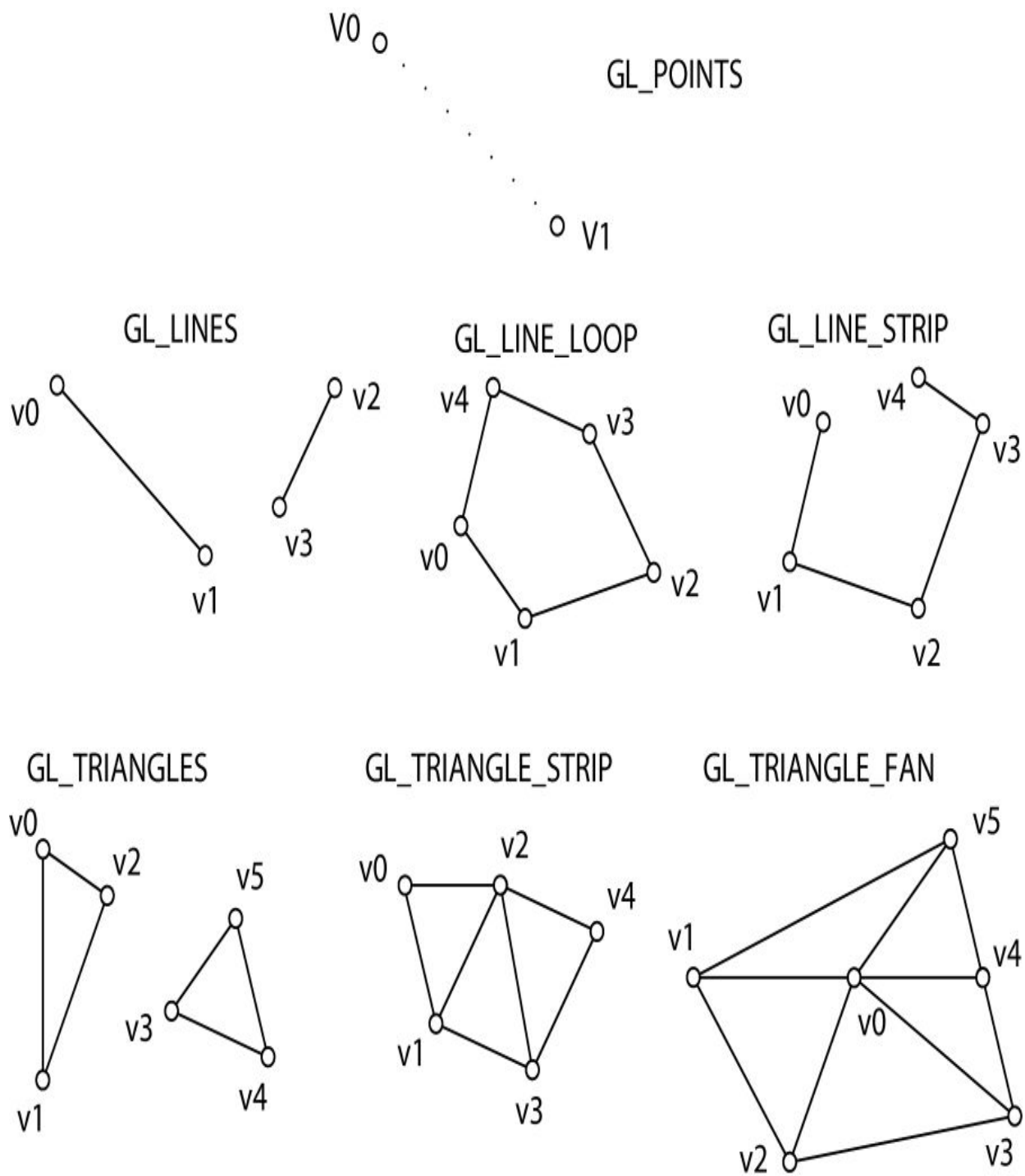
“顶点着色器”和“片段着色器”是渲染流程中的两个关键步骤。  
 “顶点着色器”负责处理顶点的坐标、颜色、纹理坐标等。  
 “片段着色器”负责处理每个片段的颜色、透明度等。

“帧缓冲区操作”包括深度测试、混和等。  
 “帧缓冲区”是存储最终渲染结果的缓冲区。  
 “帧缓冲区操作”和“帧缓冲区”是渲染流程的最后两个步骤。

## 9.2.1 顶点着色器

OpenGL  
OpenGL  
OpenGL

OpenGL  
GL\_POINTS  
GL\_LINES  
GL\_LINE\_STRIP  
GL\_LINE\_LOOP  
GL\_TRIANGLES  
GL\_TRIANGLE\_STRIP  
GL\_TRIANGLE\_FAN  
9-4  
(x, y, z)



## 9-4 OpenGL

OpenGL is a cross-platform, vendor-neutral, industry-standard API for rendering 2D and 3D computer graphics. It is designed to be efficient and flexible, allowing developers to create high-performance graphics applications. The API is implemented by various vendors, including NVIDIA, AMD, and Intel, and is supported by a wide range of hardware and software platforms.

OpenGLのレンダリングパイプライン

## 9.2.2 変換

OpenGLは、3次元空間のオブジェクトを2次元画面に投影する。この変換は、視点変換、投影変換、視口変換の3段階で行われる。視点変換は、オブジェクトを視点中心に移動し、投影変換は、オブジェクトを2次元画面に投影し、視口変換は、オブジェクトを視口の範囲に縮小する。

変換は、 $x, y, z$  の座標を  $x, y, z, w$  の座標に変換する。ここで、 $w$  は、オブジェクトの深度を表す。変換は、 $x, y, z, w$  の座標を  $x, y, z, w$  の座標に変換する。

OpenGLは、 $4 \times 4$  の変換行列を用いて、 $x, y, z, w$  の座標を  $x, y, z, w$  の座標に変換する。変換行列は、 $(x/w, y/w, z/w, 1.0)$  の座標を  $(x/w, y/w, z/w, 1.0)$  の座標に変換する。

変換行列は、 $(x, y, z, 1.0)$  の座標を  $(x + tx, y + ty, z + tz, 1.0)$  の座標に変換する。

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix}$$

OpenGLは、変換行列を用いて、オブジェクトの座標を変換する。変換行列は、 $(x, y, z, 1.0)$  の座標を  $(x + tx, y + ty, z + tz, 1.0)$  の座標に変換する。OpenGLは、変換行列を用いて、オブジェクトの座標を変換する。

[illegible][illegible]

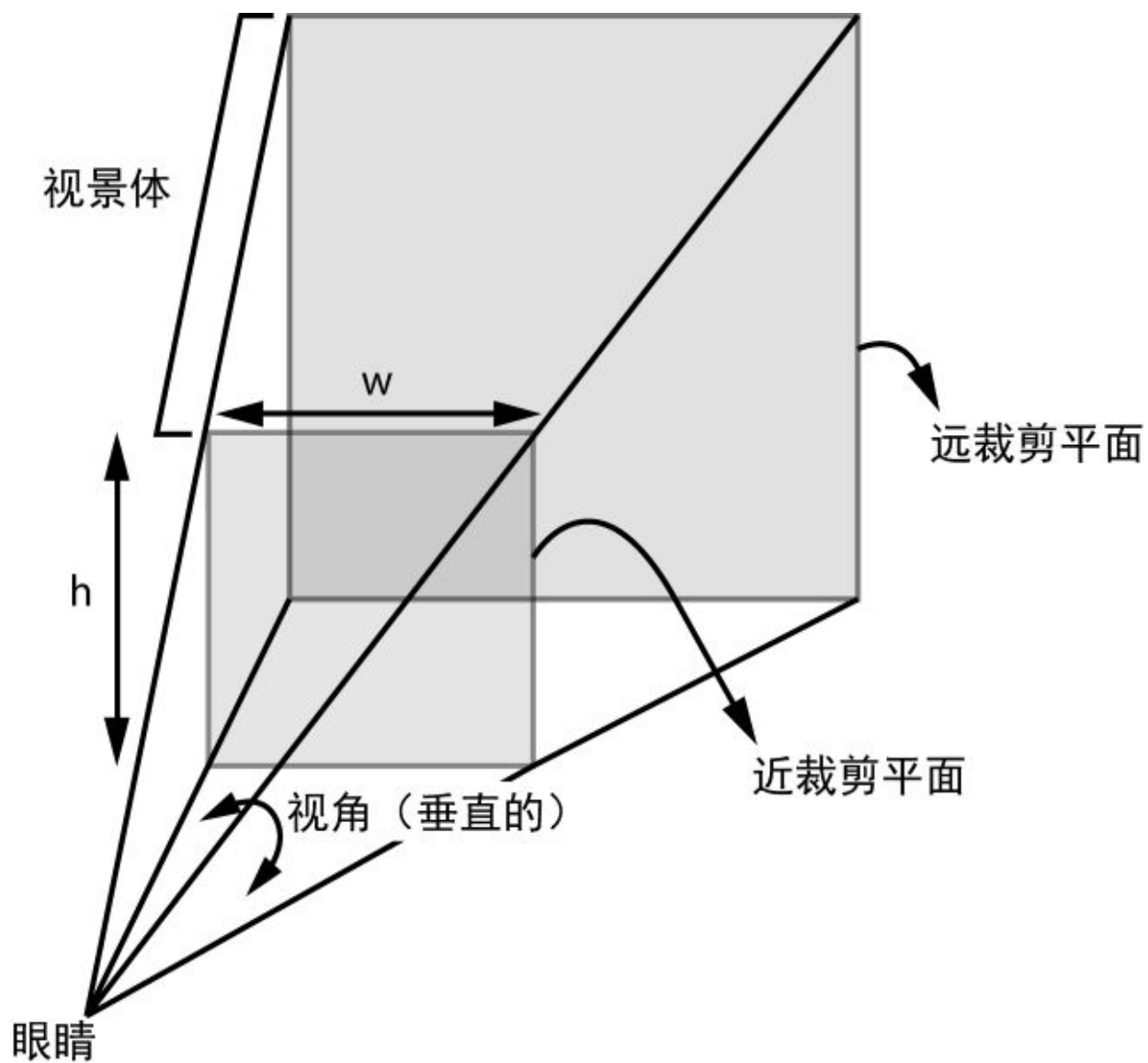


图9-5 视景体的构成

图9-5 视景体的构成

图9-5 视景体的构成

图9-5 视景体的构成

## 9.2.3 视图

# OpenGL Shader Tutorial 1: GLSL Shader

## Vertex Shader

### Vertex Shader Code

```
❶ #version 330 core

❷ in vec3 aVert;

❸ uniform mat4 uMVMatrix;
❹ uniform mat4 uPMatrix;

❺ out vec4 vCol;

void main() {
 // apply transformations
 ❻ gl_Position = uPMatrix * uMVMatrix * vec4(aVert, 1.0);

 // set color
 ❼ vCol = vec4(1.0, 0.0, 0.0, 1.0);
}
```

❶ GLSL version 3.3 in  
aVert vec3 ❷  
❸ ❹ mat4 4x4 matrix

uniform ⑤ out  
vec4 4D alpha

main() ⑥  
gl\_Position uniform  
aVert GLSL gl\_Position ⑦  
1 0 0 1

❶ #version 330 core

❷ in vec4 vCol;

❸ out vec4 fragColor;

void main() {

// use vertex color

❹ fragColor = vCol;

}



❶ 在 GLSL 中，**vec3** 类型的变量 **vCol** 可以存储一个 RGB 颜色值。在 OpenGL 中，我们通常使用 **vec3** 来存储颜色值。

在 OpenGL 中，我们通常使用 **vec3** 来存储颜色值。

❷ 在 GLSL 中，**fragColor** 是一个 **vec4** 类型的变量，用于存储片元的最终颜色。在 OpenGL 中，我们通常使用 **vec4** 来存储颜色值。

在 GPU 中，我们通常使用 **vec4** 来存储颜色值。在 OpenGL 中，我们通常使用 **vec4** 来存储颜色值。

在 Python 中，我们通常使用 **vec4** 来存储颜色值。

## 9.2.4 顶点着色器

在 OpenGL 中，顶点着色器是一个用于处理顶点的着色器。在 OpenGL 中，我们通常使用 **vec4** 来存储颜色值。在 GPU 中，我们通常使用 **vec4** 来存储颜色值。

1. 在 OpenGL 中，我们通常使用 **vec4** 来存储颜色值。

2. 在 OpenGL 中，我们通常使用 **VAO** 来存储顶点数据。

3. 创建顶点数组对象(VBO)

4. 将顶点数据写入VBO

5. 创建顶点数组对象(VAO)

6. 将VBO和VAO绑定

7. 渲染三角形

在OpenGL 3.0中，我们使用VAO来管理顶点数组。VAO是一个用于存储顶点数组的容器。它包含了指向顶点数组的指针、顶点数组的格式、顶点数组的大小等信息。在渲染时，我们只需要将VAO绑定到当前的渲染目标上，然后调用glDrawArrays()函数即可。这大大简化了顶点数组的管理工作。

## 9.2.5 三角形

在OpenGL 3.0中，我们使用GL\_TRIANGLES来渲染三角形。在9-6图中，我们使用GL\_TRIANGLES来渲染一个三角形。在代码中，我们使用glDrawArrays()函数来指定要渲染的顶点范围。在9-6图中，我们使用GL\_TRIANGLES来渲染一个三角形。在代码中，我们使用glDrawArrays()函数来指定要渲染的顶点范围。

在9-6图中，我们使用0到2的索引来指定要渲染的顶点。在代码中，我们使用glDrawArrays()函数来指定要渲染的顶点范围。在9-6图中，我们使用GL\_TRIANGLES来渲染一个三角形。在代码中，我们使用glDrawArrays()函数来指定要渲染的顶点范围。

OpenGL 1.1 版本  
 支持

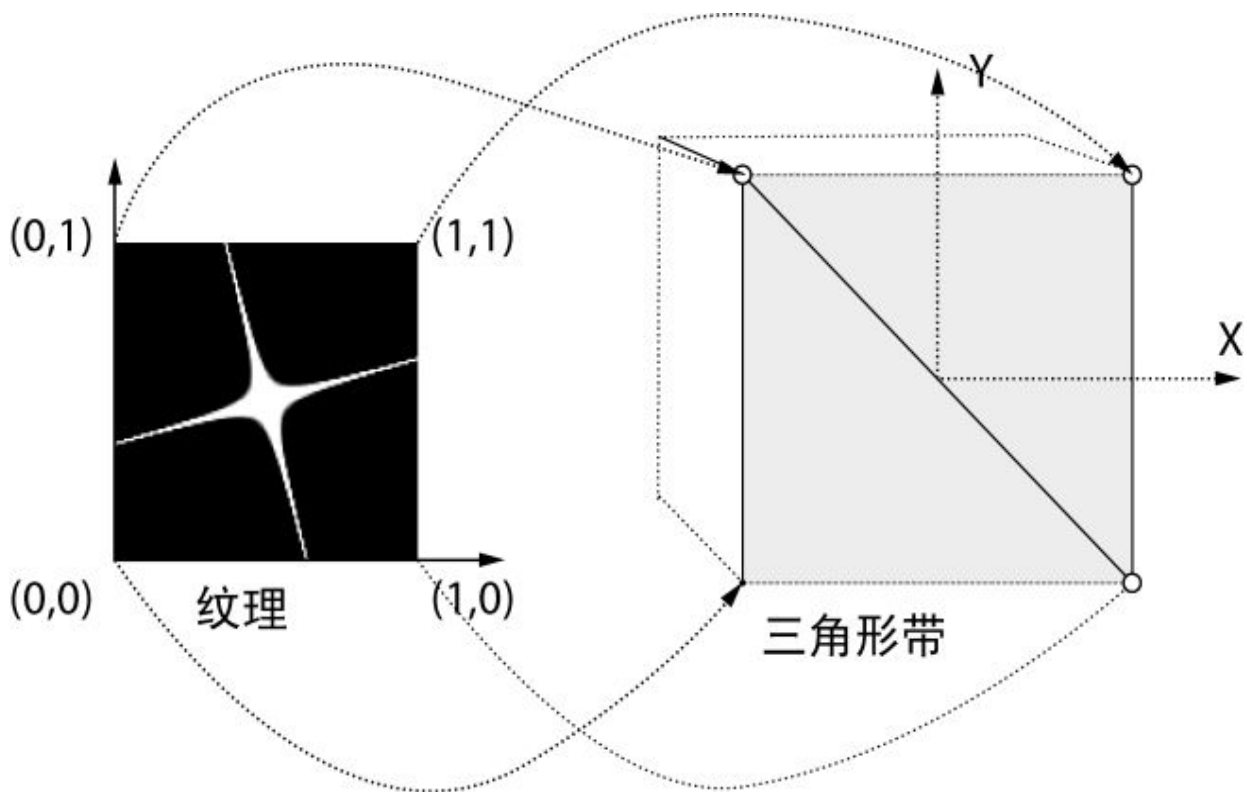


图9-6 纹理映射

## 9.2.6 OpenGL

OpenGL 是一个跨平台的图形 API，它定义了如何与图形硬件进行交互。OpenGL 本身是一个库，它提供了一系列的函数和常量，用于控制图形的渲染。OpenGL 的“OpenGL 核心”部分定义了基本的图形操作，而 OpenGL 的“OpenGL 扩展”部分则提供了更多的功能，以满足不同的需求。OpenGL 的扩展可以分为官方扩展和非官方扩展，官方扩展是由 Khronos 组织定义的，而非官方扩展则是由社区定义的。OpenGL 的扩展通常以 GL\_ 开头，后面跟着扩展的名称。

OpenGL 本身是一个库，它提供了一系列的函数和常量，用于控制图形的渲染。OpenGL 的“OpenGL 核心”部分定义了基本的图形操作，而 OpenGL 的“OpenGL 扩展”部分则提供了更多的功能，以满足不同的需求。OpenGL 的扩展通常以 GL\_ 开头，后面跟着扩展的名称。GLFW 是一个跨平台的窗口和输入库，它提供了创建窗口、管理输入和输出等功能。C 语言是 OpenGL 和 GLFW 的底层语言，它们都依赖于 C 语言来实现。

OpenGL 是一个跨平台的 API，它提供了一种标准化的方式来访问 GPU。它定义了一组函数和常量，用于控制 GPU 的渲染过程。它允许程序员使用一种高级语言（如 C 或 C++）来编写图形应用程序，而不需要直接操作 GPU 的寄存器。

Python 与 OpenGL 的交互通常通过 C 库 GLFW 来实现。Python 通过 glfw.py 模块来调用 GLFW 的函数。common 模块则提供了一些通用的函数，用于处理 OpenGL 的上下文和窗口。

## 9.3 窗口管理

PyOpenGL 是一个 Python 的 OpenGL 接口。它依赖于 GLFW 库来创建和管理窗口。numpy 库则用于处理 OpenGL 中的浮点数和整数数据。

## 9.4 渲染

OpenGL 是一个跨平台的 API，它提供了一种标准化的方式来访问 GPU。它定义了一组函数和常量，用于控制 GPU 的渲染过程。它允许程序员使用一种高级语言（如 C 或 C++）来编写图形应用程序，而不需要直接操作 GPU 的寄存器。

### 9.4.1 OpenGL 窗口

GLFW 是一个跨平台的库，用于创建和管理窗口。它提供了一种简单的方式来创建窗口，并处理窗口的事件。RenderWindow 类则是一个 Python 的 GLFW 接口，用于创建和管理 OpenGL 窗口。

以下是一个简单的 RenderWindow 类的实现：

```
class RenderWindow:
 """GLFW Rendering window class"""
 def __init__(self):
 # save current working directory
```

```

 cwd = os.getcwd()

 # initialize glfw
❶ glfw.glfwInit()

 # restore cwd
 os.chdir(cwd)

 # version hints
❷ glfw.glfwWindowHint(glfw.GlfwContextVersionMajor, 3)

 glfw.glfwWindowHint(glfw.GlfwContextVersionMinor, 3)

 glfw.glfwWindowHint(glfw.GlfwOpenGlForwardCompat, GL_TRUE)

 glfw.glfwWindowHint(glfw.GlfwOpenGlProfile,
 glfw.GlfwOpenGlCoreProfile)

 # make a window

 self.width, self.height = 640, 480

 self.aspect = self.width/float(self.height)
❸ self.win = glfw.glfwCreateWindow(self.width, self.height,
 b'simpleglfw')

```

```
make the context current
```

```
❷ glfw.glfwMakeContextCurrent(self.win)
```

❶ glfw.glfwInit() ❷ glfw.glfwCreateWindow(640, 480, glfw.glfwGetGLFWVersionString(), glfw.glfwGetGLFWVersionString(), glfw.glfwGetGLFWVersionString()) ❸ glfw.glfwSetWindowUserPointer(self) ❹ glfw.glfwMakeContextCurrent(self.win)

```
def __init__(self):
```

```
 # initialize GL
```

```
❶ glViewport(0, 0, self.width, self.height)
```

```
❷ glEnable(GL_DEPTH_TEST)
```

```
❸ glClearColor(0.5, 0.5, 0.5, 1.0)
```

❶ glfw.glfwInit() ❷ glfw.glfwCreateWindow(640, 480, glfw.glfwGetGLFWVersionString(), glfw.glfwGetGLFWVersionString(), glfw.glfwGetGLFWVersionString()) ❸ glfw.glfwSetWindowUserPointer(self) ❹ glfw.glfwMakeContextCurrent(self.win) ❺ glViewport(0, 0, self.width, self.height) ❻ glEnable(GL\_DEPTH\_TEST) ❼ glClearColor(0.5, 0.5, 0.5, 1.0)

## 9.4.2 glfw.glfwSetWindowUserPointer

```
def __init__(self):
```

```
set window callbacks
```

```
glfw.glfwSetMouseButtonCallback(self.win, self.onMouseButtonCallback)
```

n)

```
glfw.glfwSetKeyCallback(self.win, self.onKeyboard)
```

```
glfw.glfwSetWindowSizeCallback(self.win, self.onSize)
```

```
#####
#####
```

```
####
```

```
#####
```

```
def onKeyboard(self, win, key, scancode, action, mods):
```

```
 #print 'keyboard: ', win, key, scancode, action, mods
```

```
 ❶ if action == glfw.GLFW_PRESS:
```

```
 # ESC to quit
```

```
 if key == glfw.GLFW_KEY_ESCAPE:
```

```
 ❷ self.exitNow = True
```

```
 else:
```

```
 # toggle cut
```

```
 ❸ self.scene.showCircle = not self.scene.showCircle
```

```
#####onKeyboard()#####
#####key-upkey-down
#####glfw.GLFW_PRESS
keydownPRESS❶❷#####ESC
#####showCircle
#####❸
```

□□□□□□□□

□□□□□□□□□□□□□□□□

```
def onSize(self, win, width, height):
 #print 'onsize: ', win, width, height
 self.width = width
 self.height = height
 self.aspect = width/float(height)
```

❶      glfwViewport(0, 0, self.width, self.height)

□□□□□□□□□□□□□□**glfwViewport()**□□□□□□□□□□□□□□□□  
□□□□□□□□❶□□□□□□□□□□width□height□□□□□□□□  
□□□□□□□□□□aspect□□

□□□□

□□□□□□□□□□□□**GLFW**□□□□□□□□□□□□□□□□

```
def run(self):
 # initializer timer
 ❶ glfw.glfwSetTime(0)
 t = 0.0

 ❷
 while not glfw.glfwWindowShouldClose(self.win) and not self
 .exitNow:
 # update every x seconds
 ❸ currT = glfw.glfwGetTime()
```



```

 if currT - t > 0.1:
 # update time
 t = currT
 # clear

④ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

 # build projection matrix

⑤ pMatrix = glutils.perspective(45.0, self.aspect, 0.1, 100.0
)

⑥ mvMatrix = glutils.lookAt([0.0, 0.0, -2.0], [0.0, 0.0, 0.0]
,
 [0.0, 1.0, 0.0])

 # render

⑦ self.scene.render(pMatrix, mvMatrix)

 # step

⑧ self.scene.step()

⑨ glfw.glfwSwapBuffers(self.win)

 # poll for and process events

⑩ glfw.glfwPollEvents()

end

```

```
glfw.glfwTerminate()
```

```
glfw.glfwSetTime() GLFW 0 ❶
while ❷
 exitNow = True
glfw.glfwTerminate() GLFW
```

```
glfw.glfwGetTime() ❸
0.1
100
```

```
❹ glClear()
❺ glutils.py
perspective() 45 /
0.1/100.0 glutils.py
lookAt() ❻
0-2 " " (0, 1, 0) (0, 0, 0)
❷ scene render() ❸
scene.step() ❹
glfwSwapBuffers()
❺ glfwPollEvents() UI
while
```

```
'''
```

```
'''
'''
```

Windows OS  
GLFW

## 9.4.3 Scene

Scene

```
class Scene:
 """ OpenGL 3D scene class"""
 # initialization
 def __init__(self):
 # create shader
 ❶ self.program = glutils.loadShaders(strVS, strFS)

 ❷ glUseProgram(self.program)
```

Scene  
loadShaders()❶ glutils.py  
OpenGL  
OpenGL  
❷ glUseProgram(),  
“”

Python

```
self.pMatrixUniform = glGetUniformLocation(self.program, b'
uPMatrix')
```

```
self.mvMatrixUniform = glGetUniformLocation(self.program, b'
```

```
'uMVMatrix')
```

```
texture
```

```
self.tex2D = glGetUniformLocation(self.program, b'tex2D')
```

```
glGetUniformLocation() 返回一个整数，表示在着色器中变量uPMatrix、uMVMatrix、tex2D的位置。这个整数可以用于glUniformMatrix*()函数，用于在着色器中设置矩阵。
```

```
glGenVertexArrays()
```

```
glBindVertexArray()
```

```
define triangle strip vertices
```

```
❶ vertexData = numpy.array(
 [-0.5, -0.5, 0.0,
 0.5, -0.5, 0.0,
 -0.5, 0.5, 0.0,
 0.5, 0.5, 0.0], numpy.float32)
```

```
set up vertex array object (VAO)
```

```
❷ self.vao = glGenVertexArrays(1)
glBindVertexArray(self.vao)
```

```
vertices
```

```
❸ self.vertexBuffer = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)
```

```
set buffer data
```

```

④
glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData
,

 GL_STATIC_DRAW)

 # enable vertex array

⑤ glEnableVertexAttribArray(0)

 # set buffer data pointer

⑥
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, None)

 # unbind VAO

⑦ glBindVertexArray(0)

```

① 创建一个顶点数组对象 VAO，并绑定它。
   
 ② 将顶点数据（1.0, -0.5, -0.5, 0.0）写入 VBO。
   
 ③ 将 VBO 与 VAO 关联。
   
 ④ 设置顶点属性指针。

⑤ 启用顶点属性数组。
   
 ⑥ 设置顶点属性指针。
   
 ⑦ 绑定 VAO。
   
 OpenGL 渲染三角形。

# OpenGL

```
texture
```

```
self.texId = glutils.loadTexture('star.png')
```

□□□□ID□□□□□□□□

## Scene

# step

```
def step(self):
```

```
increment angle
```

```
❶ self.t = (self.t + 1) % 360
```

```
set shader angle in radians
```

```
2 glUniform1f(glGetUniformLocation(self.program, 'uTheta'),
```

```
math.radians(self.t))
```

① t % [0 360]

glUniform1f()

glGetUniformLocation()

```
uTheta = Python.math.radians()
```

[illegible]

```
render
```

```
def render(self, pMatrix, mvMatrix):
```

```
use shader
```

```

❶ glUseProgram(self.program)

 # set projection matrix

❷ glUniformMatrix4fv(self.pMatrixUniform, 1, GL_FALSE, pMatrix)

 # set modelview matrix

 glUniformMatrix4fv(self.mvMatrixUniform, 1, GL_FALSE, mvMatrix)

 # show circle?

❸ glUniform1i(glGetUniformLocation(self.program, b'showCircle'),
 self.showCircle)

 # enable texture

❹ glActiveTexture(GL_TEXTURE0)

❺ glBindTexture(GL_TEXTURE_2D, self.texId)

❻ glUniform1i(self.tex2D, 0)

 # bind VAO

❼ glBindVertexArray(self.vao)

```

```

 # draw

❸ glDrawArrays(GL_TRIANGLE_STRIP, 0, 4)

 # unbind VAO

❹ glBindVertexArray(0)

```

❶ 在 `main` 函数中，我们首先调用 `glUniformMatrix4fv()` 来更新模型视图矩阵。然后，我们调用 `glUniform1i()` 来更新 `showCircle` 变量。接着，我们调用 `glActiveTexture()` 来激活纹理 0。然后，我们调用 `sampler2D` 来声明纹理。最后，我们调用 `glDrawArrays()` 来绘制三角形。最后，我们调用 `glBindVertexArray(0)` 来解绑 VAO。

## GLSL 着色器

GLSL 着色器是用于渲染的着色器。它们通常由两个文件组成：顶点着色器和片段着色器。

```
#version 330 core
```

```
❶ layout(location = 0) in vec3 aVert;
```

```
❷ uniform mat4 uMVMatrix;

uniform mat4 uPMatrix;
```



```

 uniform float uTheta;

 ❸ out vec2 vTexCoord;

void main() {

 // rotational transform

 ❹ mat4 rot = mat4(

 vec4(cos(uTheta), sin(uTheta), 0.0, 0.0),
 vec4(-sin(uTheta), cos(uTheta), 0.0, 0.0),
 vec4(0.0, 0.0, 1.0, 0.0),
 vec4(0.0, 0.0, 0.0, 1.0)

);

 // transform vertex

 ❺
 gl_Position = uPMatrix * uMVMMatrix * rot * vec4(aVert, 1.0)
 ;

 // set texture coordinate

 ❻ vTexCoord = aVert.xy + vec2(0.5, 0.5);

}

```

❶ `layout` 声明 `aVert` 为 `vec3` 类型  
 ❷ `uniform` 声明 `uTheta` 为 `float` 类型  
 ❸ `out vec2 vTexCoord` 声明 `vTexCoord` 为 `vec2` 类型  
 ❹ `mat4` 声明 `rot` 为 `mat4` 类型  
 ❺ `gl_Position` 声明 `gl_Position` 为 `vec4` 类型  
 ❻ `vTexCoord` 声明 `vTexCoord` 为 `vec2` 类型

```

 ⑤ gl_Position ⑥
 vec4 gl_Position = vec4(
 1.0, 1.0, 1.0, 1.0);
 gl_Position.x = 0.5;
 gl_Position.y = 0.5;
 gl_Position.z = 0.5;
 gl_Position.w = 0.5;

```

```

 #version 330 core

```

```

 ① in vec4 vCol;

```

```

 in vec2 vTexCoord;

```

```

 ② uniform sampler2D tex2D;

```

```

 ③ uniform bool showCircle;

```

```

 ④ out vec4 fragColor;

```

```

void main() {

```

```

 if (showCircle) {

```

```

 // discard fragment outside circle

```

```

 ⑤ if (distance(vTexCoord, vec2(0.5, 0.5)) > 0.5) {

```

```

 discard;

```

```

 }

```

```

 else {
 ❸ fragColor = texture(tex2D, vTexCoord);
 }
 }

 else {
 ❹ fragColor = texture(tex2D, vTexCoord);
 }
}

```

❶ 在着色器中，我们使用 `texture(tex2D, vTexCoord)` 来采样纹理。这里 `tex2D` 是纹理的句柄，`vTexCoord` 是纹理坐标。在 Python 中，我们使用 `texture(tex2D, vTexCoord)` 来采样纹理。这里 `tex2D` 是纹理的句柄，`vTexCoord` 是纹理坐标。在 GLSL 中，我们使用 `texture(tex2D, vTexCoord)` 来采样纹理。这里 `tex2D` 是纹理的句柄，`vTexCoord` 是纹理坐标。

在着色器中，我们使用 `showCircle` 来显示一个圆。这里 `showCircle` 是一个函数，它接受一个半径和一个颜色作为参数。在 Python 中，我们使用 `showCircle` 来显示一个圆。这里 `showCircle` 是一个函数，它接受一个半径和一个颜色作为参数。在 GLSL 中，我们使用 `showCircle` 来显示一个圆。这里 `showCircle` 是一个函数，它接受一个半径和一个颜色作为参数。

0.5  
showCircle

## 9.5

OpenGL  
simpleglfw.py  
<https://github.com/electronut/pp/tree/master/simplegl/>  
glutils.py  
common

```
import OpenGL

from OpenGL.GL import *

import numpy, math, sys, os

import glutils

import glfw

strVS = ""

#version 330 core

layout(location = 0) in vec3 aVert;

uniform mat4 uMVMatrix;
```

```

uniform mat4 uPMatrix;

uniform float uTheta;

out vec2 vTexCoord;

void main() {

 // rotational transform

 mat4 rot = mat4(

 vec4(cos(uTheta), sin(uTheta), 0.0, 0.0),
 vec4(-sin(uTheta), cos(uTheta), 0.0, 0.0),
 vec4(0.0, 0.0, 1.0, 0.0),
 vec4(0.0, 0.0, 0.0, 1.0)

);

 // transform vertex

 gl_Position = uPMatrix * uMVMMatrix * rot * vec4(aVert, 1.0)
 ;

 // set texture coordinate

 vTexCoord = aVert.xy + vec2(0.5, 0.5);

}

"""

strFS = """

#version 330 core

```

```

in vec2 vTexCoord;

uniform sampler2D tex2D;
uniform bool showCircle;

out vec4 fragColor;

void main() {
 if (showCircle) {
 // discard fragment outside circle
 if (distance(vTexCoord, vec2(0.5, 0.5)) > 0.5) {
 discard;
 }
 else {
 fragColor = texture(tex2D, vTexCoord);
 }
 }
 else {
 fragColor = texture(tex2D, vTexCoord);
 }
}

"""

```

```

class Scene:

 """ OpenGL 3D scene class"""

 # initialization

 def __init__(self):

 # create shader

 self.program = glutils.loadShaders(strVS, strFS)

 glUseProgram(self.program)

 self.pMatrixUniform = glGetUniformLocation(self.program, b'
 uPMatrix')

 self.mvMatrixUniform = glGetUniformLocation(self.program, b'
 uMVMatrix')

 # texture

 self.tex2D = glGetUniformLocation(self.program, b'tex2D')

 # define triange strip vertices

 vertexData = numpy.array(

 [-0.5, -0.5, 0.0,

 0.5, -0.5, 0.0,

 -0.5, 0.5, 0.0,

 0.5, 0.5, 0.0], numpy.float32)

```

```

set up vertex array object (VAO)

self.vao = glGenVertexArrays(1)

glBindVertexArray(self.vao)

vertices

self.vertexBuffer = glGenBuffers(1)

glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

set buffer data

glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData
,

 GL_STATIC_DRAW)

enable vertex array

glEnableVertexAttribArray(0)

set buffer data pointer

glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, None)

unbind VAO

glBindVertexArray(0)

time

self.t = 0

texture

self.texId = glutils.loadTexture('star.png')

```



```

 # show circle?

 self.showCircle = False

step
def step(self):
 # increment angle
 self.t = (self.t + 1) % 360
 # set shader angle in radians

 glUniform1f(glGetUniformLocation(self.program, 'uTheta'),
 math.radians(self.t))

render
def render(self, pMatrix, mvMatrix):
 # use shader
 glUseProgram(self.program)

 # set projection matrix

 glUniformMatrix4fv(self.pMatrixUniform, 1, GL_FALSE, pMatrix)

 # set modelview matrix

```

```
glUniformMatrix4fv(self.mvMatrixUniform, 1, GL_FALSE, mvMatrix)
```

```
 # show circle?
```

```
glUniform1i(glGetUniformLocation(self.program, b'showCircle'),
```

```
 self.showCircle)
```

```
 # enable texture
```

```
glActiveTexture(GL_TEXTURE0)
```

```
glBindTexture(GL_TEXTURE_2D, self.texId)
```

```
glUniform1i(self.tex2D, 0)
```

```
 # bind VAO
```

```
glBindVertexArray(self.vao)
```

```
 # draw
```

```
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4)
```

```
 # unbind VAO
```

```
glBindVertexArray(0)
```

```
class RenderWindow:
```

```
 """GLFW Rendering window class"""
```

```
 def __init__(self):
```

```
save current working directory
cwd = os.getcwd()

initialize glfw - this changes cwd
glfw.glfwInit()

restore cwd
os.chdir(cwd)

version hints

glfw.glfwWindowHint(glfw.GLFW_CONTEXT_VERSION_MAJOR, 3)

glfw.glfwWindowHint(glfw.GLFW_CONTEXT_VERSION_MINOR, 3)

glfw.glfwWindowHint(glfw.GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE)

glfw.glfwWindowHint(glfw.GLFW_OPENGL_PROFILE,
 glfw.GLFW_OPENGL_CORE_PROFILE)

make a window

self.width, self.height = 640, 480

self.aspect = self.width/float(self.height)

self.win = glfw.glfwCreateWindow(self.width, self.height,
```

```
 b'simpleglfw')

make context current

glfw.glfwMakeContextCurrent(self.win)

initialize GL

glViewport(0, 0, self.width, self.height)

glEnable(GL_DEPTH_TEST)

glClearColor(0.5, 0.5, 0.5, 1.0)

set window callbacks

glfw.glfwSetMouseButtonCallback(self.win, self.onMouseButton)

glfw.glfwSetKeyCallback(self.win, self.onKeyboard)

glfw.glfwSetWindowSizeCallback(self.win, self.onSize)

create 3D

self.scene = Scene()

exit flag

self.exitNow = False
```

```
def onMouseButton(self, win, button, action, mods):
 #print 'mouse button: ', win, button, action, mods
 pass

def onKeyboard(self, win, key, scancode, action, mods):
 #print 'keyboard: ', win, key, scancode, action, mods
 if action == glfw.GLFW_PRESS:
 # ESC to quit
 if key == glfw.GLFW_KEY_ESCAPE:
 self.exitNow = True
 else:
 # toggle cut

self.scene.showCircle = not self.scene.showCircle

def onSize(self, win, width, height):
 #print 'onsize: ', win, width, height
 self.width = width
 self.height = height
 self.aspect = width/float(height)
 glViewport(0, 0, self.width, self.height)

def run(self):
```

```

 # initializer timer

 glfw.glfwSetTime(0)

 t = 0.0

while not glfw.glfwWindowShouldClose(self.win) and not self
.exitNow:

 # update every x seconds

 currT = glfw.glfwGetTime()

 if currT - t > 0.1:

 # update time

 t = currT

 # clear

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

 # build projection matrix

pMatrix = glutils.perspective(45.0, self.aspect, 0.1, 100.0
)

mvMatrix = glutils.lookAt([0.0, 0.0, -2.0], [0.0, 0.0, 0.0]
,

 [0.0, 1.0, 0.0])

 # render

 self.scene.render(pMatrix, mvMatrix)

```

```

 # step
 self.scene.step()

 glfw.glfwSwapBuffers(self.win)
 # poll for and process events
 glfw.glfwPollEvents()

 # end

 glfw.glfwTerminate()

def step(self):
 # clear
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

 # build projection matrix

 pMatrix = glutils.perspective(45.0, self.aspect, 0.1, 100.0
)

 mvMatrix = glutils.lookAt([0.0, 0.0, -2.0], [0.0, 0.0, 0.0]
 ,
 [0.0, 1.0, 0.0])

 # render
 self.scene.render(pMatrix, mvMatrix)

```

```

 # step
 self.scene.step()

 glfw.SwapBuffers(self.win)

 # poll for and process events
 glfw.PollEvents()

main() function
def main():
 print("Starting simpleglfw. "
 "Press any key to toggle cut. Press ESC to quit.")
 rw = RenderWindow()
 rw.run()

call main
if __name__ == '__main__':
 main()

```

## 9.6 OpenGL

~~~~~

```
$ python simpleglfw.py
```

~~~~~9-1~~~~~



## glutils.py OpenGL

```
def loadTexture(filename):
 """load OpenGL 2D texture from given image file"""

 ❶ img = Image.open(filename)

 ❷ imgData = numpy.array(list(img.getdata()), np.int8)

 ❸ texture = glGenTextures(1)

 ❹ glBindTexture(GL_TEXTURE_2D, texture)

 ❺ glPixelStorei(GL_UNPACK_ALIGNMENT, 1)

 ❻ glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE)

 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE)

 ❼ glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)

 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)

 ❽ glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, img.size[0], img.size[1],
 0, GL_RGBA, GL_UNSIGNED_BYTE, imgData)

 return texture
```

❶ loadTexture() Python PIL Image  
 Image ❷ Image 8 numpy ❸ OpenGL  
 OpenGL ❹ texture ❺  
 1 1 8 ❻ OpenGL  
 S T x y ❼ " " ❽

## 9.7

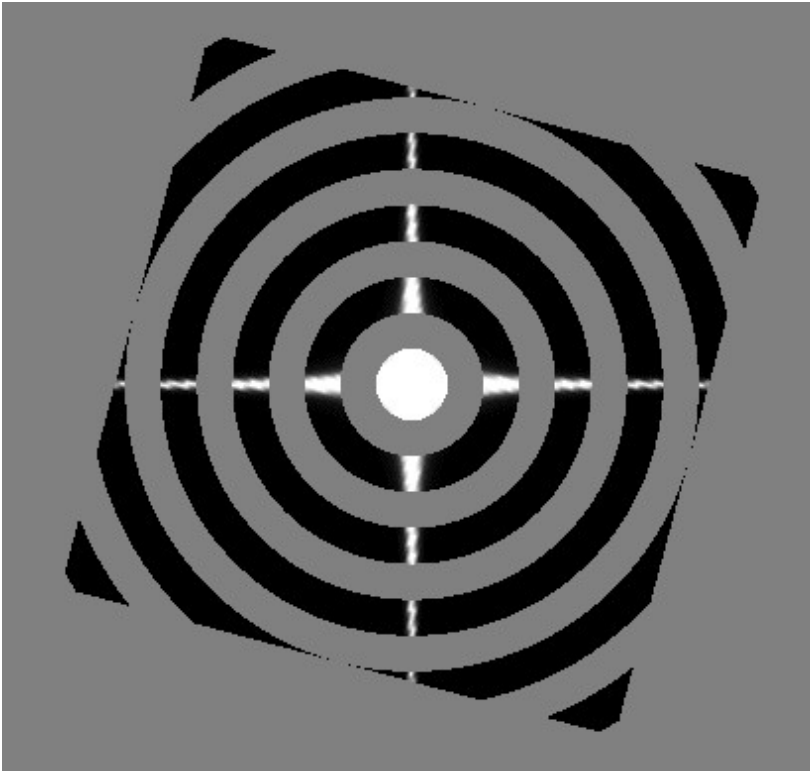
Python OpenGL

## 9.8


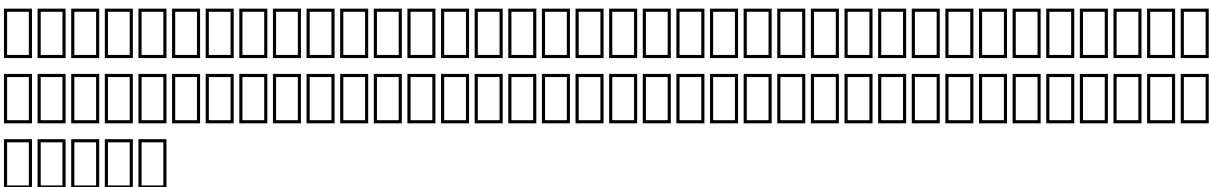
1 z 0 0 1 1 1 0 Python uniform

2

0000000000000000000000000000000000000000000000000  
0000000000000000000000000000000000000000000000000  
1.00000000glTexParameterf()00  
GL\_TEXTURE\_WRAP\_S/T00000GL\_REPEAT00  
300000000000000000000000009-700000000GLSL0  
sin()0000

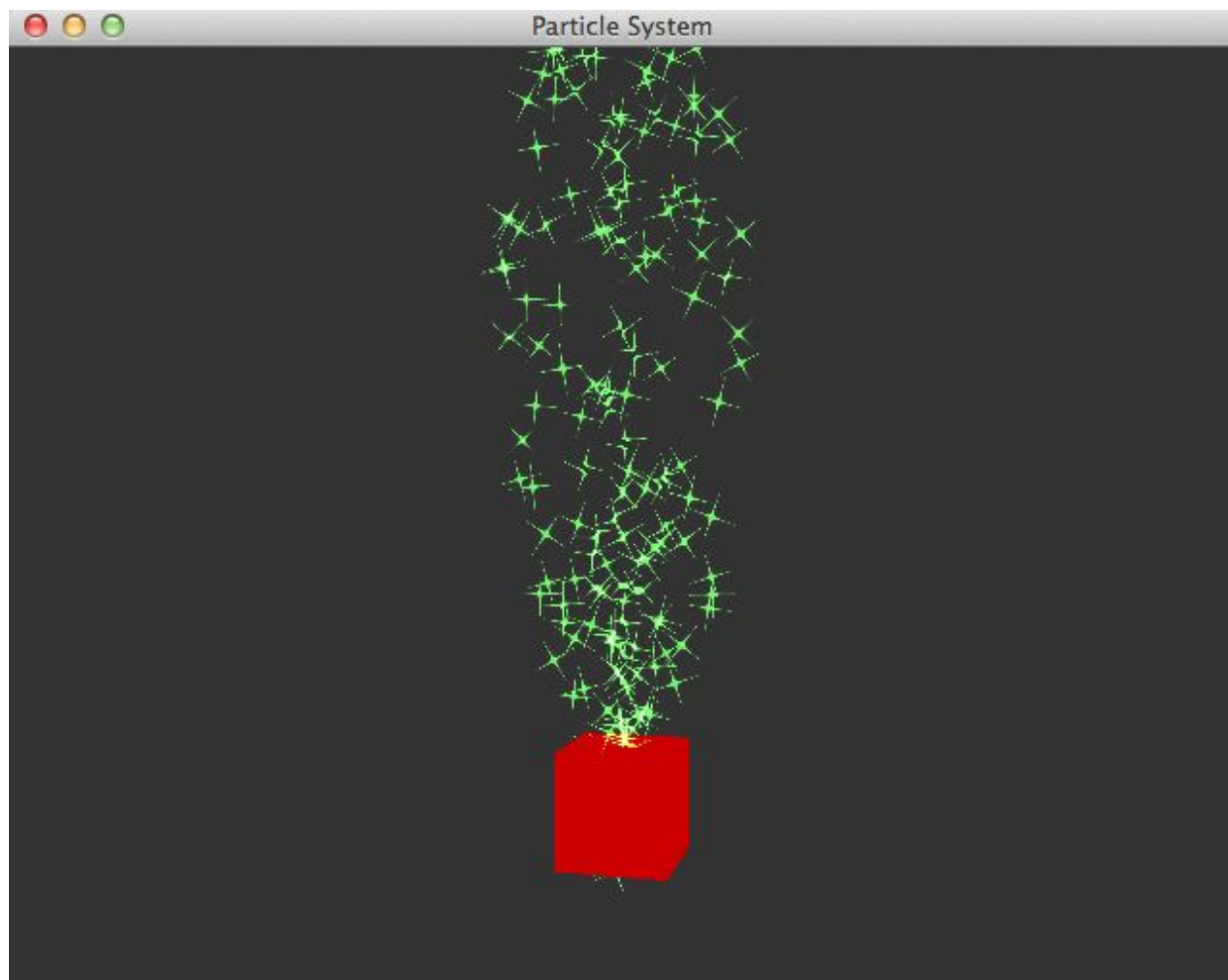


□9-7    □□□□□□□□□□□□



billboarding 10-1

GPU OpenGL



## 10-1 粒子システム

粒子システムは、3D空間に多数の粒子を生成・移動・消滅させるためのシステムである。粒子は、点、線、面、体などの形状を持つ。粒子の位置、速度、加速度、色、透明度などの属性を設定できる。粒子は、重力、風、衝突などの物理法則に従って動く。粒子は、光源の影響を受け、色や透明度が変化する。粒子は、カメラの位置や向きに応じて変形する。粒子は、時間とともに消滅する。

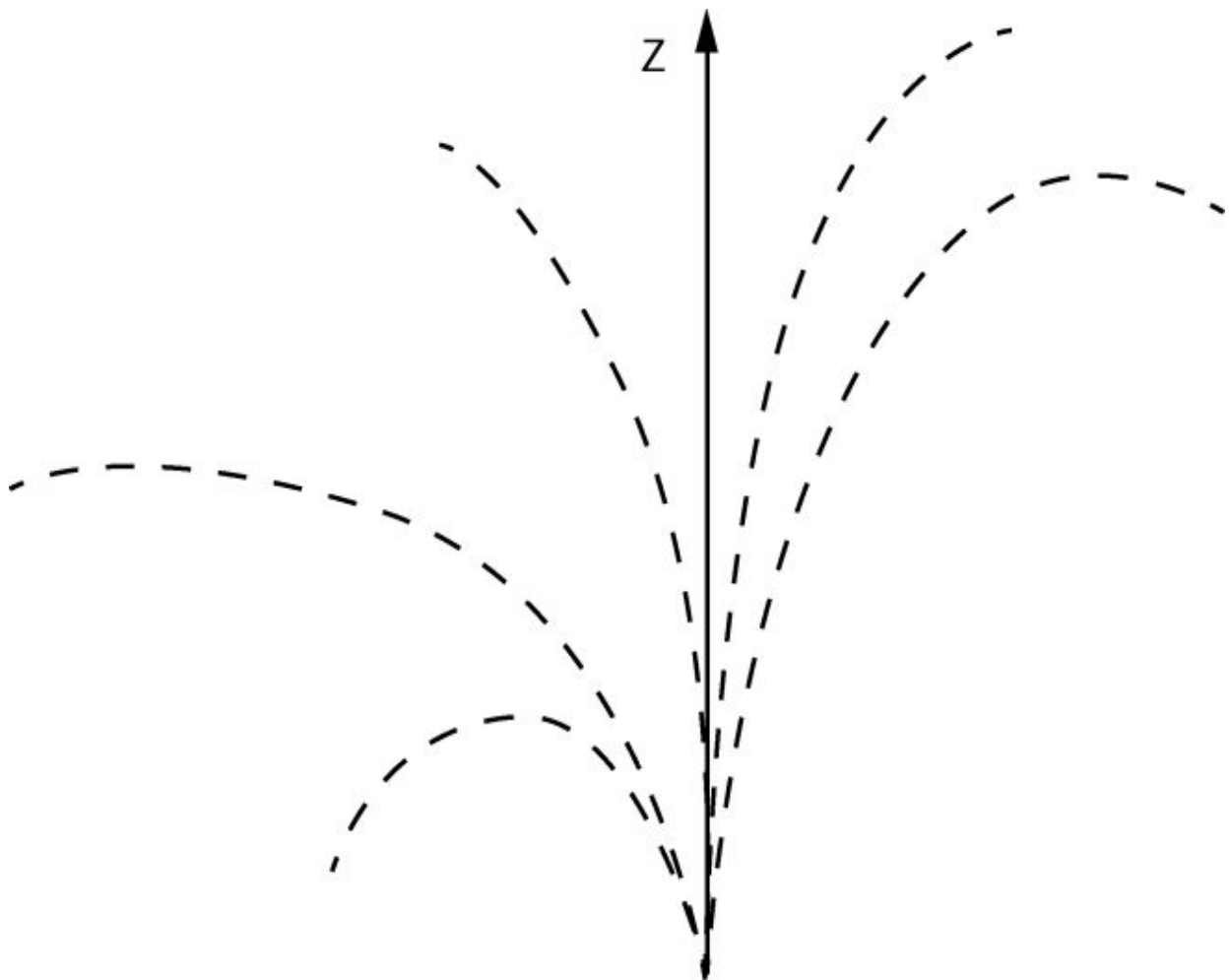
粒子システムの構成要素

- 粒子の生成・移動・消滅の制御
- GPUによる高速処理
- 物理法則のシミュレーション

- OpenGL 的 Alpha 测试
- 深度测试

## 10.1 深度测试

深度测试是 OpenGL 中用于判断物体是否可见的一种测试方法。它通过比较物体的深度值（Z 值）来决定是否渲染该物体。如果物体的深度值小于当前帧中的最大深度值，则渲染该物体；否则，跳过该物体的渲染。深度测试的精度通常为  $10^{-2}$ 。



10-2 5 深度测试的精度

深度测试的精度

- 計算機を用いて、位置、速度、加速度の時間変化を計算する
- 位置、速度、加速度の時間変化をグラフで表示する
- 位置、速度、加速度の時間変化をグラフで表示する
- 位置、速度、加速度の時間変化をグラフで表示する
- 位置、速度、加速度の時間変化をグラフで表示する

## 10.1.1 位置、速度、加速度

位置、速度、加速度の時間変化をグラフで表示する

$$P_t^i = P_0 + V_0^i t + \frac{1}{2} a t^2$$

位置  $P_t$  は時間  $t$  の関数で、 $P_0$  は初期位置、 $V_0$  は初期速度、 $a$  は加速度である。位置、速度、加速度の時間変化をグラフで表示する。

位置、速度、加速度の時間変化をグラフで表示する。位置は  $0$  から  $9.8$  の範囲で、速度は  $0$  から  $9.8$  の範囲で、加速度は  $0$  から  $9.8$  の範囲で表示する。

## 10.1.2 位置、速度、加速度

位置、速度、加速度の時間変化をグラフで表示する。位置は  $0$  から  $9.8$  の範囲で、速度は  $0$  から  $9.8$  の範囲で、加速度は  $0$  から  $9.8$  の範囲で表示する。位置は  $0$  から  $9.8$  の範囲で、速度は  $0$  から  $9.8$  の範囲で、加速度は  $0$  から  $9.8$  の範囲で表示する。

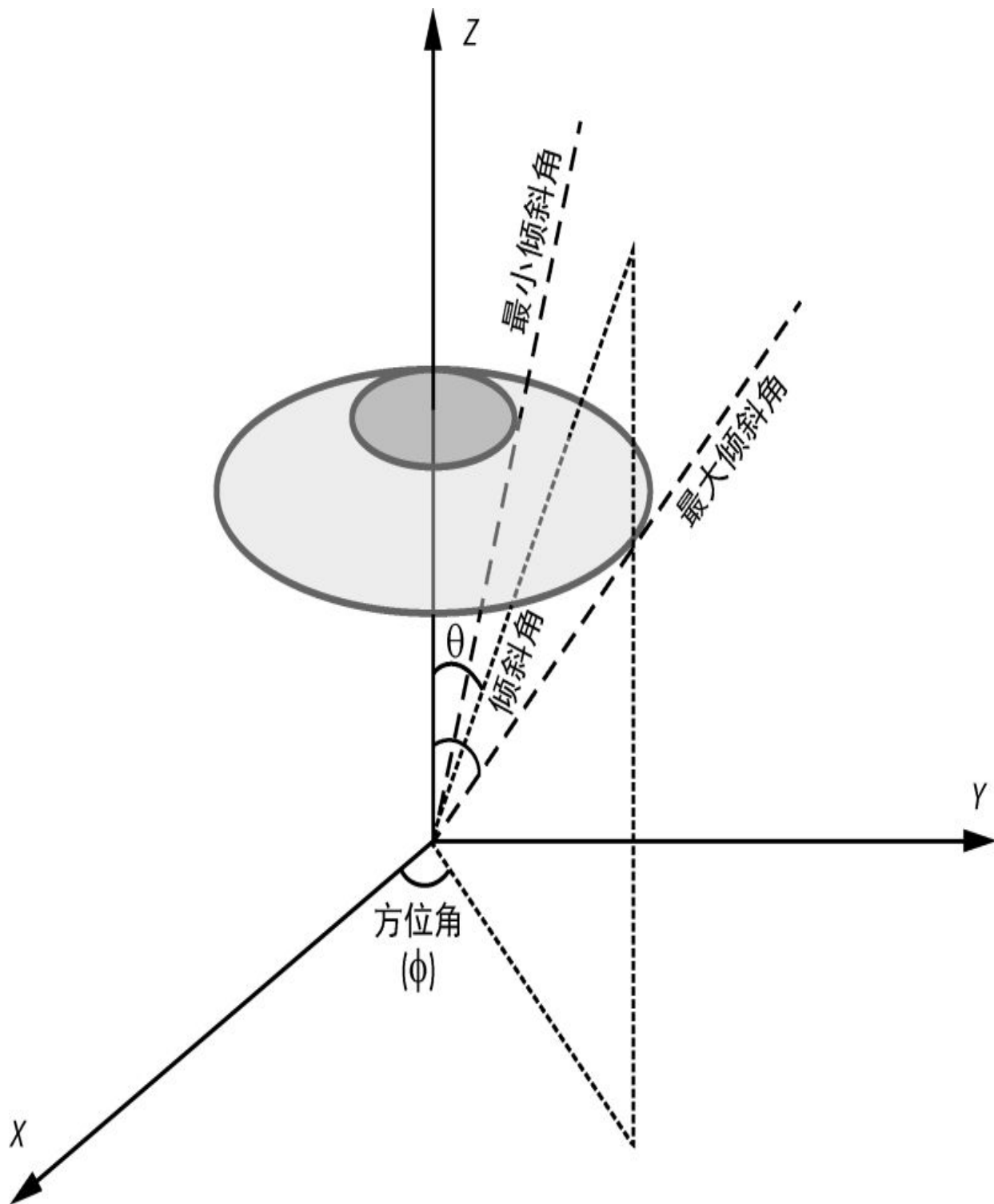


图10-3 倾斜角和方位角的定义



10-3  $\varphi$   $\times$   $\theta$   
 $z$

$$V_0^i = (1 - \alpha^2)V$$

$\alpha$  20  
1.0  $V$

$$V = (\cos(\theta) \sin(\Phi), \sin e(\theta) \sin(\Phi), \cos(\Phi))$$

[0 20] [0 360]

$$\theta = \text{random}([0, 20]), \Phi = \text{random}([0, 360])$$

10-3

10.10  
 $i$

$$t_{\text{tag}}^i = 0.05i$$

0.05  
20

OpenGL 1.1 版本中，GL\_POINTS 是点绘制的模式，即每个顶点都是一个点。

### 10.1.3 点绘制

在 OpenGL 1.1 版本中，GL\_POINTS 是点绘制的模式，即每个顶点都是一个点。

在 OpenGL 1.1 版本中，GL\_POINTS 是点绘制的模式，即每个顶点都是一个点。

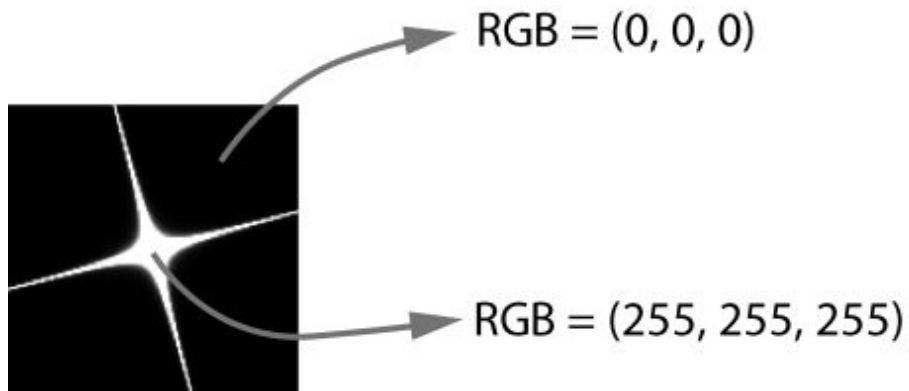
### 10.1.4 OpenGL 中的 Alpha 值

在 OpenGL 1.1 版本中，Alpha 值是点绘制的模式，即每个顶点都是一个点。

在 OpenGL 1.1 版本中，Alpha 值是点绘制的模式，即每个顶点都是一个点。

OpenGLのレンダリングは、  
10-4の図のように、

OpenGLのレンダリングは、  
alphaの値を0から1まで  
指定して、RGBの値を0から255まで  
指定して、alphaの値を0から1まで  
指定して、alphaの値を0から1まで  
指定して、10.3.5の図のように



10-4 図のように、RGBの値を0から255まで  
指定して、

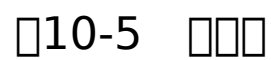


alphaの値を0から1まで  
指定して、alphaの値を0から1まで  
指定して、alphaの値を0から1まで  
指定して、alphaの値を0から1まで

## 10.1.5 図のように

図のように、  
図のように、

10-5



我们假设  $n$  是单位法向量， $v$  是速度向量， $n$  和  $v$  都是  $3 \times 1$  的列向量。

我们定义  $n = -v \times u$ ，其中  $u$  是单位向量， $u_x^2 + u_y^2 + u_z^2 = 1$ 。我们称  $u$  为“速度”向量。

我们定义  $r = u \times n$ ，其中  $r$  是单位向量， $r_x^2 + r_y^2 + r_z^2 = 1$ 。我们称  $r$  为“方向”向量。

我们定义  $u' = n \times r = n \times (u \times n)$ ，其中  $u'$  是单位向量， $u_x'^2 + u_y'^2 + u_z'^2 = 1$ 。我们称  $u'$  为“方向”向量。

我们定义  $R = \begin{bmatrix} r_x & u'_x & n_x & 0 \\ r_y & u'_y & n_y & 0 \\ r_z & u'_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ，其中  $R$  是  $4 \times 4$  的矩阵。我们称  $R$  为“变换”矩阵。

$$R = \begin{bmatrix} r_x & u'_x & n_x & 0 \\ r_y & u'_y & n_y & 0 \\ r_z & u'_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

我们假设  $R$  是正交矩阵，即  $R^T = R^{-1}$ 。

## 10.1.6 变换矩阵

GLFW  
GLFW

## 10.2

PyOpenGL Python OpenGL  
numpy

## 10.3 □□□□□□□

OpenGL 10.4

```
ParticleSystem
OpenGL
OpenGL
```

### 10.3.1 〇〇〇〇〇〇〇〇〇〇

VBO

```
create Vertex Array Object (VAO)
self.vao = glGenVertexArrays(1)
bind VAO
glBindVertexArray(self.vao)
```

□□□□□□□□□□□□□□□□□□□□□□□□

```
vertices

s = 0.2

❶ quadV = [
 -s, s, 0.0,
 -s, -s, 0.0,
 s, s, 0.0,
 s, -s, 0.0,
 s, s, 0.0,
 -s, -s, 0.0
]

❷ vertexData = numpy.array(numP*quadV, numpy.float32)
❸ self.vertexBuffer = glGenBuffers(1)
❹ glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)
❺
glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData,
 GL_STATIC_DRAW)

texture coordinates

❻ quadT = [
 0.0, 1.0,
 0.0, 0.0,
 1.0, 1.0,
```

```

 1.0, 0.0,
 1.0, 1.0,
 0.0, 0.0
]

 tcData = numpy.array(numP*quadT, numpy.float32)

 self.tcBuffer = glGenBuffers(1)

 glBindBuffer(GL_ARRAY_BUFFER, self.tcBuffer)

 glBufferData(GL_ARRAY_BUFFER, 4*len(tcData), tcData, GL_STATIC_DRAW)

```

❶ 0.4 GL\_TRIANGLES ❷ numP numpy

9 ❸ VBO ❹ ❺ 4 \* len(vertexData) vertexData 4

❻ VBO

## 10.3.2



```

time lags

❶
timeData = numpy.repeat(0.005*numpy.arange(numP, dtype=numpy.float32),
 4)

self.timeBuffer = glGenBuffers(1)

glBindBuffer(GL_ARRAY_BUFFER, self.timeBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(timeData), timeData,
 GL_STATIC_DRAW)

```

❶ numpy.arange() returns an array of values from 0 to numP-1. The values are multiplied by 0.005 and repeated 4 times. The resulting array is then used to create a VBO.

### 10.3.3 Particle Velocities

The particle velocities are stored in a list. The list is initialized with zeros. The velocities are then set to random values between 0 and 1.

```

velocities

velocities = []

cone angle

❶ coneAngle = math.radians(20.0)

set up particle velocities

for i in range(numP):

```

```

 # inclination
❷ angleRatio = random.random()

 a = angleRatio*coneAngle

 # azimuth
❸ t = random.random()*(2.0*math.pi)

 # get velocity on sphere
❹ vx = math.sin(a)*math.cos(t)
 vy = math.sin(a)*math.sin(t)
 vz = math.cos(a)

 # speed decreases with angle
❺ speed = 15.0*(1.0 - angleRatio*angleRatio)

 # add a set of calculated velocities
❻ velocities += 6*[speed*vx, speed*vy, speed*vz]

 # set up velocity vertex buffer
 self.velBuffer = glGenBuffers(1)
 glBindBuffer(GL_ARRAY_BUFFER, self.velBuffer)
❽ velData = numpy.array(velocities, numpy.float32)

glBufferData(GL_ARRAY_BUFFER, 4*len(velData), velData, GL_S
TATIC_DRAW)

```

❶ `math.radians()` converts degrees to radians. 10.1.1

② 随机数生成器。在C++中，我们使用 `random.random()` 函数来生成一个在 `[0, 1]` 范围内的随机浮点数。然后，我们将其乘以 `2.0 * math.pi` 来得到 `0` 到 `2π` 之间的随机角度。

④ 我们使用 `random.random()` 函数来生成一个在 `[0, 1]` 范围内的随机浮点数。⑤ 我们使用 `random.random()` 函数来生成一个在 `[0, 1]` 范围内的随机浮点数。⑥ 我们使用 `random.random()` 函数来生成一个在 `[0, 1]` 范围内的随机浮点数。⑦ 我们使用 `random.random()` 函数来生成一个在 `[0, 1]` 范围内的随机浮点数。Python 中，我们使用 `numpy` 库来生成随机数。VBO 是顶点缓冲对象。⑨ 我们使用 `random.random()` 函数来生成一个在 `[0, 1]` 范围内的随机浮点数。

## 10.3.4 顶点着色器

顶点着色器是 OpenGL 渲染管线中的一个重要组成部分。它负责计算每个顶点的颜色、位置和其他属性。在 C++ 中，我们使用 GLSL 语言来编写顶点着色器。以下是一个简单的顶点着色器示例：

```
#version 330 core

in vec3 aVel;
in vec3 aVert;
in float aTime0;
in vec2 aTexCoord;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;
uniform mat4 bMatrix;
```

```
out vec4 vCol;

out vec2 vTexCoord;
```

```
main()
```

```

 // apply a twist

 float PI = 3.14159265358979323846264;

 5 float theta = mod(100.0*length(aVel)*dt, 360.0)*PI/180.0;

 6 mat4 rot = mat4(vec4(cos(theta), sin(theta), 0.0, 0.0),
 vec4(-sin(theta), cos(theta), 0.0, 0.0),
 vec4(0.0, 0.0, 1.0, 0.0),
 vec4(0.0, 0.0, 0.0, 1.0));

 // apply billboard matrix

 7 vec4 pos2 = bMatrix*rot*vec4(aVert, 1.0);

 // calculate position

 8 vec3 newPos = pos2.xyz + uPos + aVel*dt + 0.5*accel*dt*dt;

 // apply transformations

 9 gl_Position = uPMatrix * uMVMatrix * vec4(newPos, 1.0);

 }

 // set color

 10 vCol = vec4(uColor.rgb, alpha);

 // set texture coordinates

 vTexCoord = aTexCoord;

 }

```

❶ 将 `alpha` 限制在 `[0, 1]` 范围内

❷ 将 `alpha` 限制在 `[0, 1]` 范围内

❸ 将 `alpha` 限制在 `[0, 1]` 范围内

$$R_{\theta, z} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0.0 & 0.0 \\ -\sin(\theta) & \cos(\theta) & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

❹ 将 `alpha` 限制在 `[0, 1]` 范围内

❺ 将 `alpha` 限制在 `[0, 1]` 范围内

## 10.3.5 纹理着色器

纹理着色器负责为每个像素计算颜色。

```
#version 330 core

uniform sampler2D uSampler;

in vec4 vCol;
in vec2 vTexCoord;
out vec4 fragColor;

void main() {

 // get the texture color

 ❶ vec4 texCol = texture2D(uSampler, vec2(vTexCoord.s, vTexCoord.t));

 // multiply texture color by set vertex color; use the vertex color alpha

 ❷ fragColor = vec4(texCol.rgb*vCol.rgb, vCol.a);
}
```

❶ 在GLSL中，`texture2D()`函数用于从纹理中获取颜色。它接受两个参数：纹理采样器和纹理坐标。返回值为一个包含RGBA分量的vec4向量。

❷ 将纹理颜色与顶点颜色相乘，并设置alpha值。最后，将结果赋值给`fragColor`，以便传递给下一个着色阶段。

`ParticleSystem.restart()`函数用于重置粒子系统，包括alpha值。

## 10.3.6 相机

相机模型参数

1 相机中心/光心

2 相机焦距

3 相机主点

4 相机畸变系数，通常用 `uniform` 存储

5 相机视场角

6 相机近裁剪面

7 相机远裁剪面

8 相机 OpenGL 相机模型

9 相机视锥体

相机模型参数

相机模型参数

相机模型参数

```
N = camera.eye - camera.center
```

❶ `N /= numpy.linalg.norm(N)`

```
U = camera.up
```



```

 U /= numpy.linalg.norm(U)

 R = numpy.cross(U, N)

 U2 = numpy.cross(N, R)

❷ bMatrix = numpy.array([R[0], U2[0], N[0], 0.0,

 R[1], U2[1], N[1], 0.0,

 R[2], U2[2], N[2], 0.0,

 0.0, 0.0, 0.0, 1.0], numpy.float32)

❸ glUniformMatrix4fv(self.bMatrixU, 1, GL_TRUE, bMatrix)

```

我们使用 `numpy.linalg.norm()` 来计算向量的模长，然后将其归一化。❶  
 我们使用 `numpy.cross()` 来计算两个向量的叉积。❷  
 我们使用 `numpy.float32` 来指定数组的数据类型。❸

我们使用 `alpha` 来控制透明度。

我们使用 `OpenGL` 来渲染纹理。

```

 # enable texture

❶ glActiveTexture(GL_TEXTURE0)

❷ glBindTexture(GL_TEXTURE_2D, self.texid)

❸ glUniform1i(self.samplerU, 0)

```



```

Create a VAO and VBO
Create a VAO and VBO
Create a VAO and VBO
Create a VAO and VBO

```

⑤ Create OpenGL buffers for vertex and index data  
 alpha ⑥ Create OpenGL buffers for vertex and index data  
 VAO ⑦ Create OpenGL buffers for vertex and index data  
 ⑧ Create OpenGL buffers for vertex and index data  
 buffers

## 10.3.7 Camera

Create a Camera class for OpenGL rendering

```

a simple camera class

class Camera:

 """helper class for viewing"""

 ❶ def __init__(self, eye, center, up):

 self.r = 10.0

 self.theta = 0

 self.eye = numpy.array(eye, numpy.float32)

 self.center = numpy.array(center, numpy.float32)

 self.up = numpy.array(up, numpy.float32)

 def rotate(self):

 """rotate eye by one step"""

```

```

❷ self.theta = (self.theta + 1) % 360

recalculate eye

❸ self.eye = self.center + numpy.array([
 self.r*math.cos(math.radians(self.theta)),
 self.r*math.sin(math.radians(self.theta)),
 0.0], numpy.float32)

```

3. Camera rotate()

❶ camera.rotate()

❷

❸

❸

$(r \cos(\theta), r \sin(\theta))$

$x$  center

## 10.4

https://github.com/electronut/pp/tree/master/ particle-system/ps.py

```

import sys, random, math

import OpenGL

```

```
from OpenGL.GL import *
```

```
import numpy
```

```
import glutils
```

```
strVS = ""
```

```
#version 330 core
```

```
in vec3 aVel;
```

```
in vec3 aVert;
```

```
in float aTime0;
```

```
in vec2 aTexCoord;
```

```
uniform mat4 uMVMatrix;
```

```
uniform mat4 uPMatrix;
```

```
uniform mat4 bMatrix;
```

```
uniform float uTime;
```

```
uniform float uLifeTime;
```

```
uniform vec4 uColor;
```

```
uniform vec3 uPos;
```

```
out vec4 vCol;
```

```
out vec2 vTexCoord;
```

```

void main() {

 // set position

 float dt = uTime - aTime0;

 float alpha = clamp(1.0 - 2.0*dt/uLifeTime, 0.0, 1.0);

 if(dt < 0.0 || dt > uLifeTime || alpha < 0.01) {

 // out of sight!

 gl_Position = vec4(0.0, 0.0, -1000.0, 1.0);

 }

 else {

 // calculate new position

 vec3 accel = vec3(0.0, 0.0, -9.8);

 // apply a twist

 float PI = 3.14159265358979323846264;

 float theta = mod(100.0*length(aVel)*dt, 360.0)*PI/180.0;

 mat4 rot = mat4(vec4(cos(theta), sin(theta), 0.0, 0.0),
 vec4(-sin(theta), cos(theta), 0.0, 0.0),
 vec4(0.0, 0.0, 1.0, 0.0),
 vec4(0.0, 0.0, 0.0, 1.0));

 // apply billboard matrix

 vec4 pos2 = bMatrix*rot*vec4(aVert, 1.0);

 // calculate position
 }
}

```

```

vec3 newPos = pos2.xyz + uPos + aVel*dt + 0.5*accel*dt*dt;

 // apply transformations

gl_Position = uPMatrix * uMVMMatrix * vec4(newPos, 1.0);

 }

 // set color

 vCol = vec4(uColor.rgb, alpha);

 // set tex coords

 vTexCoord = aTexCoord;

}

"""

```

```

strFS = ""

```

```

#version 330 core

```

```

uniform sampler2D uSampler;

```

```

in vec4 vCol;

```

```

in vec2 vTexCoord;

```

```

out vec4 fragColor;

```

```

void main() {

```

```

 // get texture color

```

```
vec4 texCol = texture(uSampler, vec2(vTexCoord.s, vTexCoord
.t));
```

```
 // multiply by set vertex color; use the vertex color al
pha
```

```
 fragColor = vec4(texCol.rgb*vCol.rgb, vCol.a);
```

```
}
```

```
"""
```

```
a simple camera class
```

```
class Camera:
```

```
 """helper class for viewing"""
```

```
 def __init__(self, eye, center, up):
```

```
 self.r = 10.0
```

```
 self.theta = 0
```

```
 self.eye = numpy.array(eye, numpy.float32)
```

```
 self.center = numpy.array(center, numpy.float32)
```

```
 self.up = numpy.array(up, numpy.float32)
```

```
 def rotate(self):
```

```
 """rotate eye by one step"""
```

```
 self.theta = (self.theta + 1) % 360
```

```
 # recalculate eye
```

```
 self.eye = self.center + numpy.array([
```

```
 self.r*math.cos(math.radians(self.theta)),
```



```

 self.r*math.sin(math.radians(self.theta)),
 0.0], numpy.float32)

particle system class
class ParticleSystem:

 # initialization
 def __init__(self, numP):
 # number of particles
 self.numP = numP

 # time variable
 self.t = 0.0

 self.lifeTime = 5.0

 self.startPos = numpy.array([0.0, 0.0, 0.5])

 # load texture
 self.texid = glutils.loadTexture('star.png')

 # create shader
 self.program = glutils.loadShaders(strVS, strFS)
 glUseProgram(self.program)

 # set sampler
 texLoc = glGetUniformLocation(self.program, b"uTex")

```

```
glUniform1i(texLoc, 0)
```

```
uniforms
```

```
self.timeU = glGetUniformLocation(self.program, b"uTime")
```

```
self.lifeTimeU = glGetUniformLocation(self.program, b"uLife
Time")
```

```
self.pMatrixUniform = glGetUniformLocation(self.program, b'
uPMatrix')
```

```
self.mvMatrixUniform = glGetUniformLocation(self.program, b
"uMVMatrix")
```

```
self.bMatrixU = glGetUniformLocation(self.program, b"bMatri
x")
```

```
self.colorU = glGetUniformLocation(self.program, b"uColor")
```

```
self.samplerU = glGetUniformLocation(self.program, b"uSampl
er")
```

```
self.posU = glGetUniformLocation(self.program, b"uPos")
```

```
attributes
```

```
self.vertIndex = glGetAttribLocation(self.program, b"aVert"
)
```

```
self.texIndex = glGetAttribLocation(self.program, b"aTexCoo
```

```
rd")
```

```
self.time0Index = glGetAttribLocation(self.program, b"aTime
0")
```

```
self.velIndex = glGetAttribLocation(self.program, b"aVel")
```

```
 # render flags
```

```
 self.enableBillboard = True
```

```
 self.disableDepthMask = True
```

```
 self.enableBlend = True
```

```
 # which texture to use
```

```
 self.useStarTexture = True
```

```
 # restart - first time
```

```
 self.restart(numP)
```

```
step
```

```
def step(self):
```

```
 # increment time
```

```
 self.t += 0.01
```

```
restart particle system
```

```
def restart(self, numP):
```

```
set number of particles
self.numP = numP

time variables
self.t = 0.0
self.lifeTime = 5.0

color
self.col0 = numpy.array([random.random(), random.random(),
 random.random(), 1.0])

create Vertex Arrays Object (VAO)
self.vao = glGenVertexArrays(1)
bind VAO
glBindVertexArray(self.vao)

create attribute arrays and vertex buffers:

vertices
s = 0.2
quadV = [
 -s, s, 0.0,
```

```

 -s, -s, 0.0,
 s, s, 0.0,
 s, -s, 0.0,
 s, s, 0.0,
 -s, -s, 0.0
]

 vertexData = numpy.array(numP*quadV, numpy.float32)
 self.vertexBuffer = glGenBuffers(1)
 glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

 glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData
 ,

 GL_STATIC_DRAW)

texture coordinates
quadT = [
 0.0, 1.0,
 0.0, 0.0,
 1.0, 1.0,
 1.0, 0.0,
 1.0, 1.0,
 0.0, 0.0
]

 tcData = numpy.array(numP*quadT, numpy.float32)

```

```

 self.tcBuffer = glGenBuffers(1)

 glBindBuffer(GL_ARRAY_BUFFER, self.tcBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(tcData), tcData, GL_STATIC_DRAW)

 # time lags

timeData = numpy.repeat(0.005*numpy.arange(numP, dtype=numpy.float32),

 4)

 self.timeBuffer = glGenBuffers(1)

 glBindBuffer(GL_ARRAY_BUFFER, self.timeBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(timeData), timeData,

 GL_STATIC_DRAW)

 # velocities

 velocities = []

 # cone angle

 coneAngle = math.radians(20.0)

 # set up particle velocities

 for i in range(numP):

 # inclination

 angleRatio = random.random()

 a = angleRatio*coneAngle

```

```

 # azimuth
 t = random.random()*(2.0*math.pi)

 # get velocity on sphere
 vx = math.sin(a)*math.cos(t)
 vy = math.sin(a)*math.sin(t)
 vz = math.cos(a)

 # speed decreases with angle
 speed = 15.0*(1.0 - angleRatio*angleRatio)

 # add a set of calculated velocities
 velocities += 6*[speed*vx, speed*vy, speed*vz]

set up velocity vertex buffer
self.velBuffer = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, self.velBuffer)
velData = numpy.array(velocities, numpy.float32)

glBufferData(GL_ARRAY_BUFFER, 4*len(velData), velData, GL_STATIC_DRAW)

enable arrays
glEnableVertexAttribArray(self.vertIndex)
glEnableVertexAttribArray(self.texIndex)
glEnableVertexAttribArray(self.time0Index)
glEnableVertexAttribArray(self.velIndex)

```

```

 # set buffers

 glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

 glVertexAttribPointer(self.vertIndex, 3, GL_FLOAT, GL_FALSE,
 0, None)

 glBindBuffer(GL_ARRAY_BUFFER, self.tcBuffer)

 glVertexAttribPointer(self.texIndex, 2, GL_FLOAT, GL_FALSE,
 0, None)

 glBindBuffer(GL_ARRAY_BUFFER, self.velBuffer)

 glVertexAttribPointer(self.velIndex, 3, GL_FLOAT, GL_FALSE,
 0, None)

 glBindBuffer(GL_ARRAY_BUFFER, self.timeBuffer)

 glVertexAttribPointer(self.time0Index, 1, GL_FLOAT, GL_FALSE,
 0, None)

 # unbind VAO

 glBindVertexArray(0)

 # render the particle system

 def render(self, pMatrix, mvMatrix, camera):

```



```

use shader

glUseProgram(self.program)

set projection matrix

glUniformMatrix4fv(self.pMatrixUniform, 1, GL_FALSE, pMatrix)

set modelview matrix

glUniformMatrix4fv(self.mvMatrixUniform, 1, GL_FALSE, mvMatrix)

set up a billboard matrix to keep quad aligned to view direction
if self.enableBillboard:
 N = camera.eye - camera.center
 N /= numpy.linalg.norm(N)
 U = camera.up
 U /= numpy.linalg.norm(U)
 R = numpy.cross(U, N)
 U2 = numpy.cross(N, R)
 bMatrix = numpy.array([R[0], U2[0], N[0], 0.0,
 R[1], U2[1], N[1], 0.0,
 R[2], U2[2], N[2], 0.0,

```

```

0.0, 0.0, 0.0, 1.0], numpy.float32)

glUniformMatrix4fv(self.bMatrixU, 1, GL_TRUE, bMatrix)

 else:

 # identity matrix

 bMatrix = numpy.array([1.0, 0.0, 0.0, 0.0,
 0.0, 1.0, 0.0, 0.0,
 0.0, 0.0, 1.0, 0.0,
 0.0, 0.0, 0.0, 1.0], numpy.float32)

glUniformMatrix4fv(self.bMatrixU, 1, GL_FALSE, bMatrix)

 # set start position
 glUniform3fv(self.posU, 1, self.startPos)

 # set time
 glUniform1f(self.timeU, self.t)

 #set lifetime
 glUniform1f(self.lifeTimeU, self.lifeTime)

 # set color
 glUniform4fv(self.colorU, 1, self.col0)

 # enable texture
 glActiveTexture(GL_TEXTURE0)
 glBindTexture(GL_TEXTURE_2D, self.texid)

```

```
glUniform1i(self.samplerU, 0)

turn depth mask off
if self.disableDepthMask:
 glDepthMask(GL_FALSE)

enable blending
if self.enableBlend:
 glBlendFunc(GL_SRC_ALPHA, GL_ONE)
 glEnable(GL_BLEND)

bind VAO
glBindVertexArray(self.vao)

draw
glDrawArrays(GL_TRIANGLES, 0, 6*self.numP)

unbind VAO
glBindVertexArray(0)

disable blend
if self.enableBlend:
 glDisable(GL_BLEND)
```

```
turn depth mask on

if self.disableDepthMask:

 glDepthMask(GL_TRUE)

disable texture

glBindTexture(GL_TEXTURE_2D, 0)
```

```


```

## 10.5 10.5

```


```

```
import sys, random, math

import OpenGL

from OpenGL.GL import *

import numpy

import glutils
```

```
strVS = ""
```

```
#version 330 core
```

```
in vec3 aVert;
```

```
uniform mat4 uMVMatrix;
```

```
uniform mat4 uPMatrix;
```

```
out vec4 vCol;
```

```
void main() {
```

```
 // apply transformations
```

```
 gl_Position = uPMatrix * uMVMatrix * vec4(aVert, 1.0);
```

```
 // set color
```

```
 vCol = vec4(0.8, 0.0, 0.0, 1.0);
```

```
}
```

```
"""
```

```
strFS = """
```

```
#version 330 core
```

```
in vec4 vCol;
```

```
out vec4 fragColor;
```

```
void main() {
```

```
 // use vertex color
```

```
 fragColor = vCol;
```

```
}
```

```
"""
```

```

class Box:
 def __init__(self, side):
 self.side = side

 # load shaders

 self.program = glutils.loadShaders(strVS, strFS)
 glUseProgram(self.program)

 s = side/2.0
 vertices = [
 -s, s, -s,
 -s, -s, -s,
 s, s, -s,
 s, -s, -s,
 s, s, s,
 -s, -s, s,

 -s, s, s,
 -s, -s, s,
 s, s, s,
 s, -s, s,
 s, s, s,
 s, s, s,

```

-S, -S, S,

-S, -S, S,

-S, -S, -S,

S, -S, S,

S, -S, -S,

S, -S, S,

-S, -S, -S,

-S, S, S,

-S, S, -S,

S, S, S,

S, S, -S,

S, S, S,

-S, S, -S,

-S, -S, S,

-S, -S, -S,

-S, S, S,

-S, S, -S,

-S, S, S,

-S, -S, -S,

```
S, -S, S,

S, -S, -S,

S, S, S,

S, S, -S,

S, S, S,

S, -S, -S

]
```

```
set up vertex array object (VAO)
self.vao = glGenVertexArrays(1)
glBindVertexArray(self.vao)

set up VBOs
vertexData = numpy.array(vertices, numpy.float32)
self.vertexBuffer = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData
,

 GL_STATIC_DRAW)

#enable arrays

self.vertIndex = glGetAttribLocation(self.program, "aVert")
glEnableVertexAttribArray(self.vertIndex)
```



```

 # set buffers

 glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glVertexAttribPointer(self.vertIndex, 3, GL_FLOAT, GL_FALSE
, 0, None)

 # unbind VAO

 glBindVertexArray(0)

def render(self, pMatrix, mvMatrix):

 # use shader

 glUseProgram(self.program)

 # set projection matrix

 glUniformMatrix4fv(glGetUniformLocation(self.program, 'uPMa
trix'),

 1, GL_FALSE, pMatrix)

 # set modelview matrix

 glUniformMatrix4fv(glGetUniformLocation(self.program, 'uMVM
atrix'),

 1, GL_FALSE, mvMatrix)

 # bind VAO

 glBindVertexArray(self.vao)

 # draw

```

```
glDrawArrays(GL_TRIANGLES, 0, 36)
```

```
unbind VAO
```

```
glBindVertexArray(0)
```

```

#####9#####
```

## 10.6 粒子系统

粒子系统程序名为psmain.py使用GLFW窗口系统  
粒子系统程序名为psmain.py使用GLFW窗口系统10.7

```
class PSMaker:
```

```
 """GLFW Rendering window class for Particle System"""
```

```
 def __init__(self):
```

```
❶ self.camera = Camera([15.0, 0.0, 2.5],
 [0.0, 0.0, 2.5],
 [0.0, 0.0, 1.0])
```

```
 self.aspect = 1.0
```

```
 self.numP = 300
```

```
 self.t = 0
```

```
 # flag to rotate camera view
```

```
 self.rotate = True
```

```
 # save current working directory
```

```

 cwd = os.getcwd()

 # initialize glfw; this changes cwd
❷ glfw.glfwInit()

 # restore cwd
 os.chdir(cwd)

 # version hints

glfw.glfwWindowHint(glfw.GlfwContextVersionMajor, 3)

glfw.glfwWindowHint(glfw.GlfwContextVersionMinor, 3)

glfw.glfwWindowHint(glfw.GlfwOpenGlForwardCompat, GL_TRUE)

glfw.glfwWindowHint(glfw.GlfwOpenGlProfile,
 glfw.GlfwOpenGlCoreProfile)

 # make a window
 self.width, self.height = 640, 480
 self.aspect = self.width/float(self.height)

self.win = glfw.glfwCreateWindow(self.width, self.height,
 b"Particle System")

```

```

make context current
glfw.glfwMakeContextCurrent(self.win)

initialize GL
glViewport(0, 0, self.width, self.height)
glEnable(GL_DEPTH_TEST)
glClearColor(0.2, 0.2, 0.2, 1.0)

set window callbacks

glfw.glfwSetMouseButtonCallback(self.win, self.onMouseButton)

glfw.glfwSetKeyCallback(self.win, self.onKeyboard)
glfw.glfwSetWindowSizeCallback(self.win, self.onSize)

create 3D
❸ self.psys = ParticleSystem(self.numP)
❹ self.box = Box(1.0)

exit flag
❺ self.exitNow = False

```

PSMaker GLFW Camera ❶

OpenGL② GLFW③ ParticleSystem④ Box⑤  
GLFW⑤

## 10.6.1 □□□□□□□□

[illegible]

`step()`

```
def step(self):
 # increment time
 ❶ self.t += 10
 ❷ self.psys.step()
 # rotate eye
 if self.rotate:
 ❸ self.camera.rotate()
 # restart every 5 seconds
 if not int(self.t) % 5000:
 ❹ self.psys.restart(self.numP)
```

ParticleSystem.step()

□□□□□❸□□□□□□□5□□□5000□□□□□□□□□□❹□□□□□  
□.

## 10.6.2 □□□□□□

□□□□□□GLFW□□□□□□□□□□

```
❶ def onKeyboard(self, win, key, scancode, action, mods):
 #print 'keyboard: ', win, key, scancode, action, mods
 if action == glfw.GLFW_PRESS:
 # ESC to quit
 if key == glfw.GLFW_KEY_ESCAPE:
 self.exitNow = True
 elif key == glfw.GLFW_KEY_R:
 self.rotate = not self.rotate
 elif key == glfw.GLFW_KEY_B:
 # toggle billboard

self.psys.enableBillboard = not self.psys.enableBillboard
 elif key == glfw.GLFW_KEY_D:
 # toggle depth mask

self.psys.disableDepthMask = not self.psys.disableDepthMask
 elif key == glfw.GLFW_KEY_T:
 # toggle transparency
```

```
self.psys.enableBlend = not self.psys.enableBlend
```

❶

```
glDisable(GL_BLEND)
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)
```

## 10.6.3 时间管理

使用GLFW时间管理函数

```
def run(self):
 # initializer timer
 glfw.SetTime(0)
 t = 0.0
```

❶

```
while not glfw.glfwWindowShouldClose(self.win) and not self.f.exitNow:
```

```
 # update every x seconds
```

❷

```
 currT = glfw.glfwGetTime()
```

```
 if currT - t > 0.01:
```

```
 # update time
```

```
 t = currT
```

```
 # clear
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
```

```
 # render
```

```
pMatrix = glutils.perspective(100.0, self.aspect, 0.1, 100.0)
```

```
 # modelview matrix
```

```
mvMatrix = glutils.lookAt(self.camera.eye, self.camera.center,
```

```
 self.camera.up)
```

```
 # draw nontransparent object first
```

```
③ self.box.render(pMatrix, mvMatrix)
```

```
 # render
```

```
④ self.psys.render(pMatrix, mvMatrix, self.camera)
```

```
 # step
```

```
⑤ self.step()
```

```
 glfw.glfwSwapBuffers(self.win)
```

```
 # poll for and process events
```

```
 glfw.glfwPollEvents()
```

```
end
```

```
glfw.glfwTerminate()
```



pygame.quit() 9 pygame.quit() pygame.quit() ① pygame.quit() exit() pygame.quit()  
pygame.quit() while pygame.quit() ② pygame.quit() pygame.quit()  
pygame.quit() pygame.quit() pygame.quit() 0.1 pygame.quit() pygame.quit()  
pygame.quit() ③ pygame.quit() ④ pygame.quit() pygame.quit() pygame.quit()  
pygame.quit() pygame.quit() pygame.quit() pygame.quit() pygame.quit() pygame.quit()  
pygame.quit() ⑤ pygame.quit() pygame.quit() pygame.quit() pygame.quit()

## 10.7 pygame pygame

pygame.psmain.py pygame pygame  
<https://github.com/electronut/pp/tree/master/particle-system/> pygame

```
import sys, os, math, numpy

import OpenGL

from OpenGL.GL import *

import numpy

from ps import ParticleSystem, Camera

from box import Box

import glutils

import glfw

class PSMaker:

 """GLFW Rendering window class for Particle System"""

 def __init__(self):
```

```
self.camera = Camera([15.0, 0.0, 2.5],
 [0.0, 0.0, 2.5],
 [0.0, 0.0, 1.0])

self.aspect = 1.0

self.numP = 300

self.t = 0

flag to rotate camera view
self.rotate = True

save current working directory
cwd = os.getcwd()

initialize glfw; this changes cwd
glfw.glfwInit()

restore cwd
os.chdir(cwd)

version hints

glfw.glfwWindowHint(glfw.GLFW_CONTEXT_VERSION_MAJOR, 3)

glfw.glfwWindowHint(glfw.GLFW_CONTEXT_VERSION_MINOR, 3)
```

```
glfw.glfwWindowHint(glfw.GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE)
```

```
 glfw.glfwWindowHint(glfw.GLFW_OPENGL_PROFILE,
 glfw.GLFW_OPENGL_CORE_PROFILE)
```

```
make a window
```

```
self.width, self.height = 640, 480
```

```
self.aspect = self.width/float(self.height)
```

```
self.win = glfw.glfwCreateWindow(self.width, self.height,
 b"Particle System")
```

```
make context current
```

```
glfw.glfwMakeContextCurrent(self.win)
```

```
initialize GL
```

```
glViewport(0, 0, self.width, self.height)
```

```
glEnable(GL_DEPTH_TEST)
```

```
glClearColor(0.2, 0.2, 0.2,1.0)
```

```
set window callbacks
```

```
glfw.glfwSetMouseButtonCallback(self.win, self.onMouseButton)
```

```
glfw.glfwSetKeyCallback(self.win, self.onKeyboard)
```

```
glfw.glfwSetWindowSizeCallback(self.win, self.onSize)
```

```

create 3D

self.psys = ParticleSystem(self.numP)

self.box = Box(1.0)

exit flag

self.exitNow = False

def onMouseButton(self, win, button, action, mods):

 #print 'mouse button: ', win, button, action, mods

 pass

def onKeyboard(self, win, key, scancode, action, mods):

 #print 'keyboard: ', win, key, scancode, action, mods

 if action == glfw.GLFW_PRESS:

 # ESC to quit

 if key == glfw.GLFW_KEY_ESCAPE:

 self.exitNow = True

 elif key == glfw.GLFW_KEY_R:

 self.rotate = not self.rotate

 elif key == glfw.GLFW_KEY_B:

```

```

 # toggle billboard
self.psys.enableBillboard = not self.psys.enableBillboard

 elif key == glfw.GLFW_KEY_D:
 # toggle depth mask

self.psys.disableDepthMask = not self.psys.disableDepthMask

 elif key == glfw.GLFW_KEY_T:
 # toggle transparency

self.psys.enableBlend = not self.psys.enableBlend

def onSize(self, win, width, height):
 #print 'onsize: ', win, width, height
 self.width = width
 self.height = height
 self.aspect = width/float(height)
 glViewport(0, 0, self.width, self.height)

def step(self):
 # increment time
 self.t += 10
 self.psys.step()
 # rotate eye

```

```

 if self.rotate:
 self.camera.rotate()

 # restart every 5 seconds
 if not int(self.t) % 5000:
 self.psys.restart(self.numP)

 def run(self):
 # initializer timer
 glfw.glfwSetTime(0)
 t = 0.0

 while not glfw.glfwWindowShouldClose(self.win) and not self
 .exitNow:

 # update every x seconds
 currT = glfw.glfwGetTime()

 if currT - t > 0.01:
 # update time
 t = currT

 # clear

 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

 # render

```

```
pMatrix = glutils.perspective(100.0, self.aspect, 0.1, 100.0)

 # modelview matrix

mvMatrix = glutils.lookAt(self.camera.eye, self.camera.center,

 self.camera.up)

 # draw nontransparent object first
 self.box.render(pMatrix, mvMatrix)

 # render

self.psys.render(pMatrix, mvMatrix, self.camera)

 # step
 self.step()

 glfw.glfwSwapBuffers(self.win)
 # poll for and process events
 glfw.glfwPollEvents()

end

glfw.glfwTerminate()
```





1 在着色器中定义一个全局变量

2 在着色器中定义一个全局变量

3 在着色器中定义一个全局变量  
在着色器中定义一个全局变量 GLSL noise() 函数  
在着色器中

4 在着色器中定义一个全局变量  
在着色器中定义一个全局变量  $z =$   
0.0 在着色器中定义一个全局变量  $z$



MRI CT

MRI CT  $N_x \times N_y \times N_z$   
 $N_z$  “ ”  $N_x \times N_y$

“volume ray casting”  
GPU OpenGL  
GLSL  
GPU

$x, y, z$  的坐标值。在 OpenGL 中，我们使用 `glVertex3f` 函数来指定顶点的坐标。这个函数接受三个浮点参数，分别代表 x、y 和 z 坐标。

在 OpenGL 中，我们使用 `glVertex3f` 函数来指定顶点的坐标。

- GLSL 是 GPU 上的着色语言
- 用于编写顶点着色器和片段着色器
- 使用 `glShaderSource` 函数来指定着色器的源代码
- `numpy` 是一个用于科学计算的库

## 11.1 顶点着色器

在 OpenGL 中，顶点着色器是负责处理顶点坐标的着色器。它接收从顶点数组中提取的顶点坐标，并对其进行变换，以生成最终的顶点位置。顶点着色器的输出是一个 `vec3` 类型的向量，表示顶点在 3D 空间中的位置。

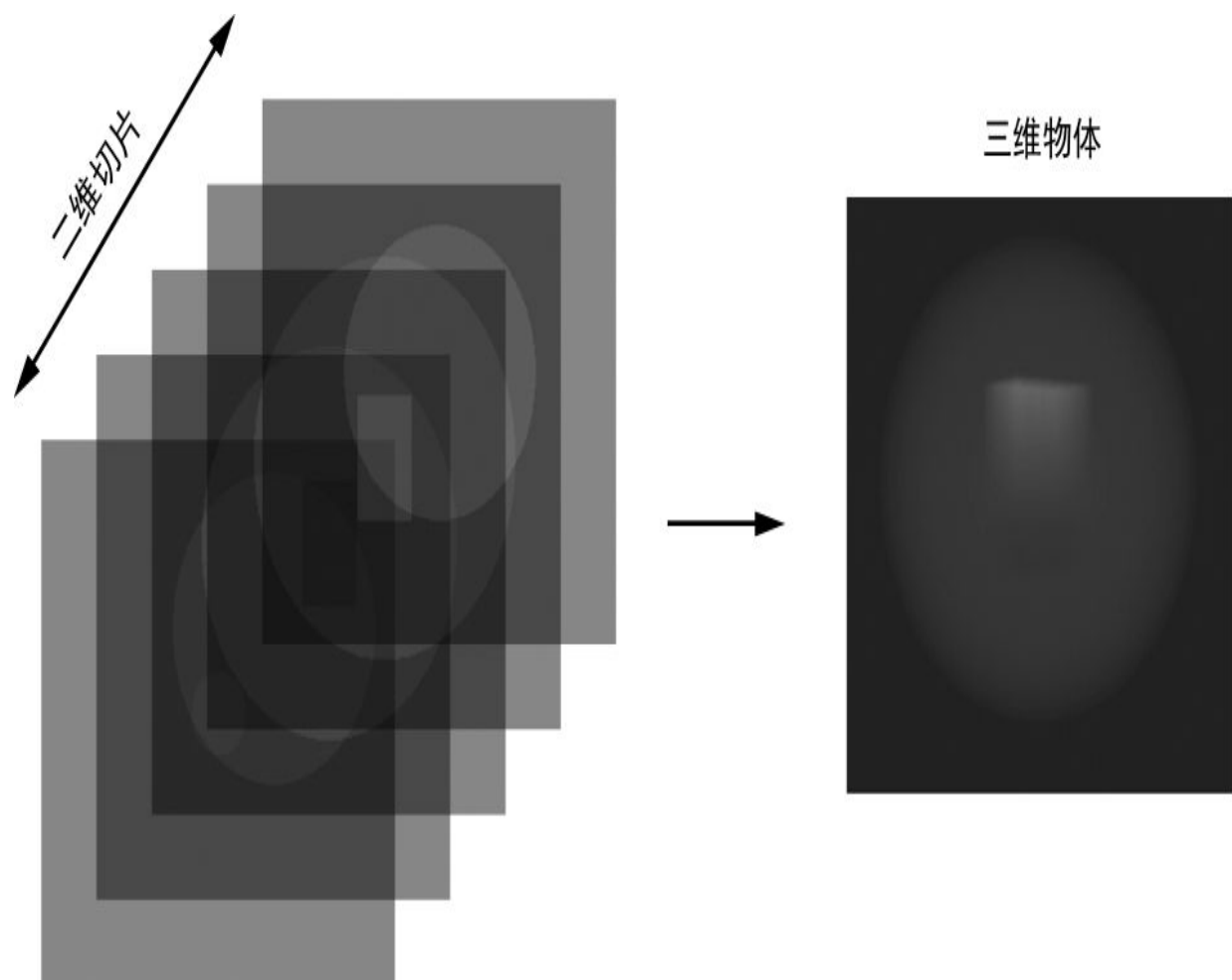
在 OpenGL 中，顶点着色器的源代码通常使用 GLSL 编写。我们使用 `glShaderSource` 函数来指定着色器的源代码。这个函数接受一个字符串数组，其中每个字符串代表着色器源代码的一部分。在顶点着色器中，我们通常使用 `vec3` 类型的变量来存储顶点的坐标。

在 OpenGL 中，我们使用 `glVertex3f` 函数来指定顶点的坐标。这个函数接受三个浮点参数，分别代表 x、y 和 z 坐标。在 2003 年，Kruger 和 Westermann 在他们的书中 [1] 中详细讨论了 OpenGL 的顶点着色器。

## 11.1.1 体素

体素数据可以从Stanford Volume Data Archive [\[2\]](#) 中获取。体素数据通常以TIFF格式存储，CT、MRI等数据。OpenGL 11-1

9 OpenGL 11-1 s t p



11-1

## 11.1.2

11-2

11-2 OpenGL 9

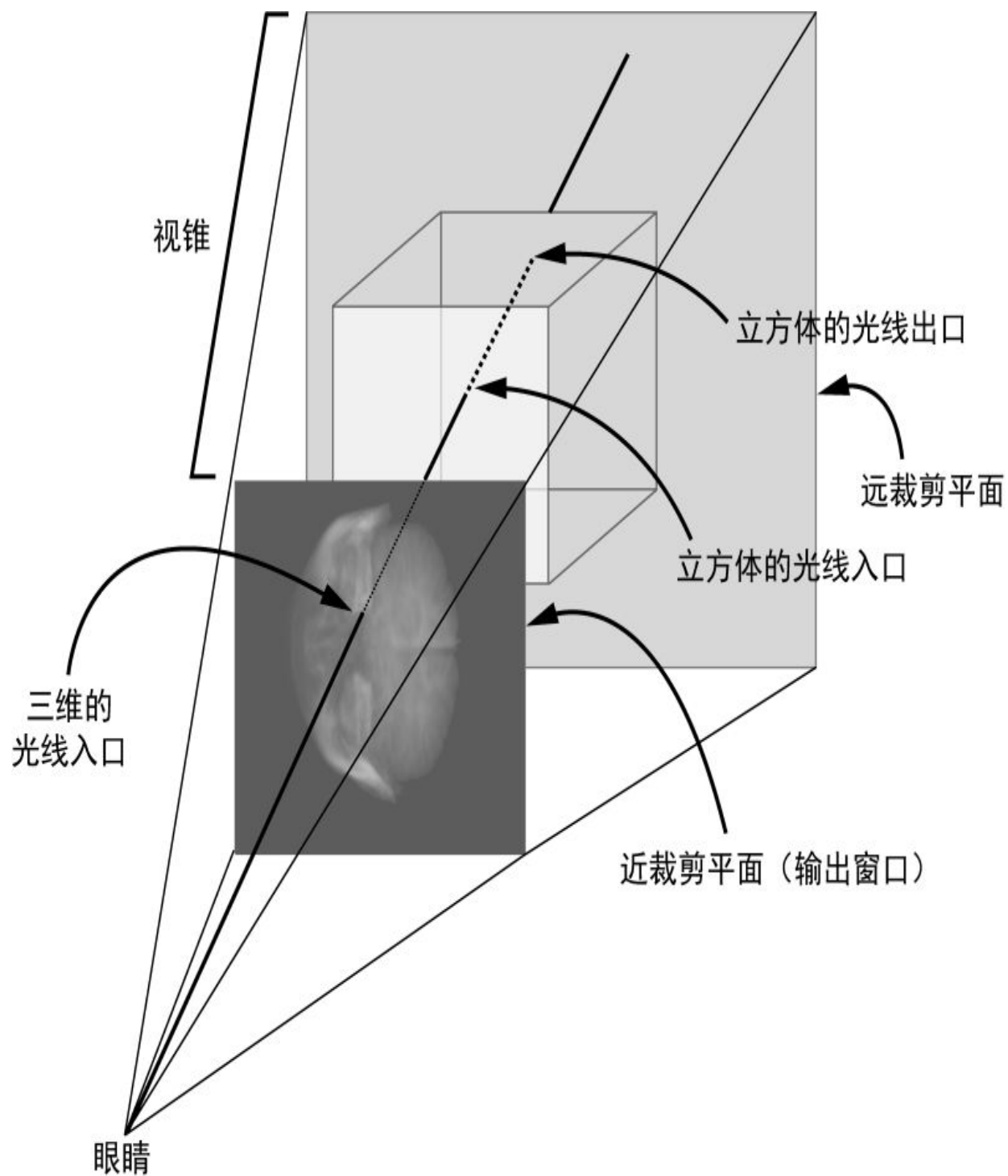
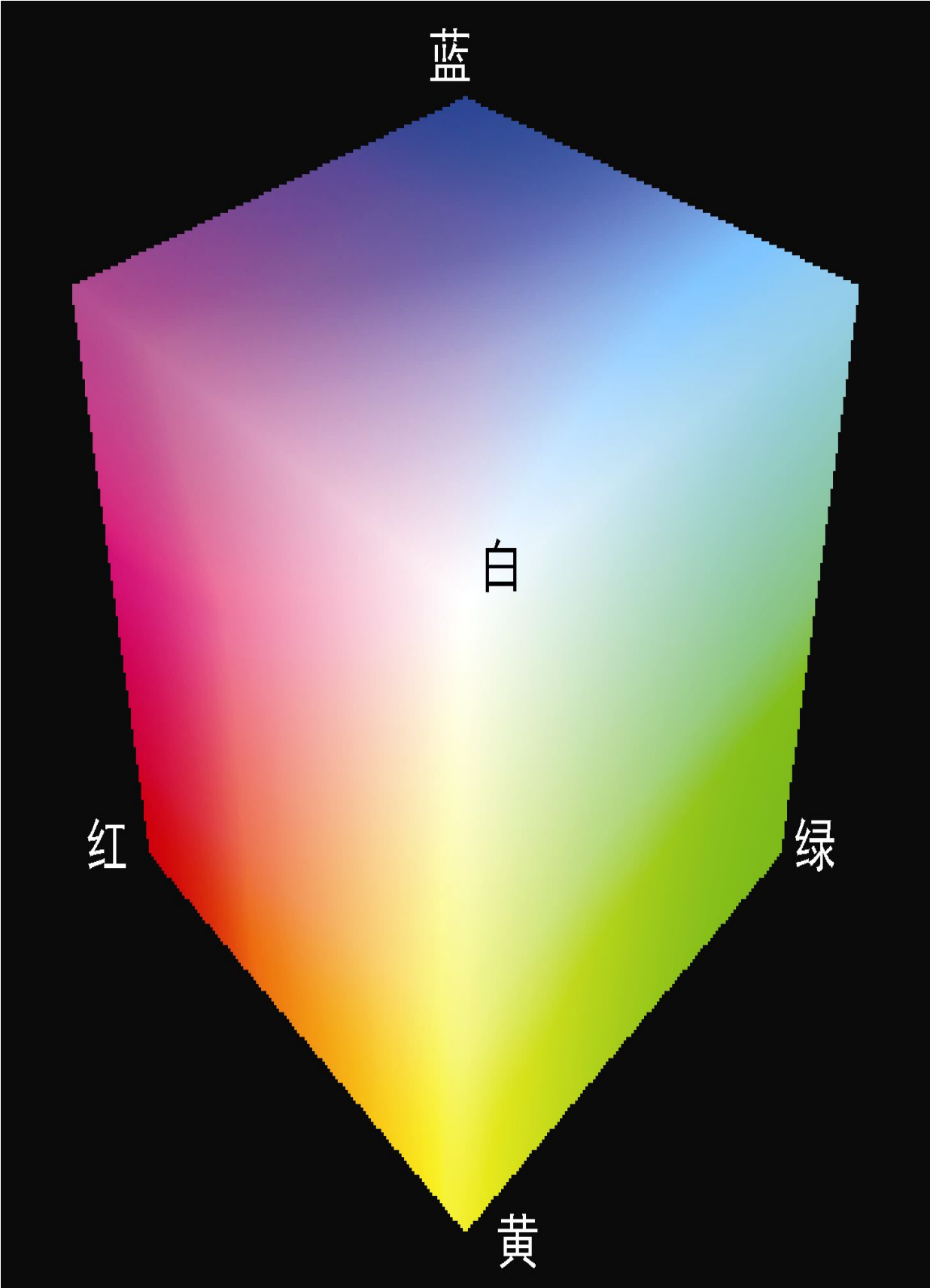


图11-2 视图的构成

图11-2 视图的构成

图11-2 视图的构成

[illegible]





□11-3    □□□□□



```

OpenGL
8(r, g, b)r gb[0
255]32(r, g, b)r gb[0.01.0]
2550001.00.00.0

```

OpenGL GL\_TRIANGLES 6  
OpenGL 11-4 A  
11-4 B OpenGL

```

11-4 B 11-4 A (r, g,
back (r, g, b) front
(r, g, b) 11-
4 C
(r, g, b) 11-4 C
rx, ry, rz

```

OpenGL  
FBO  
Texture

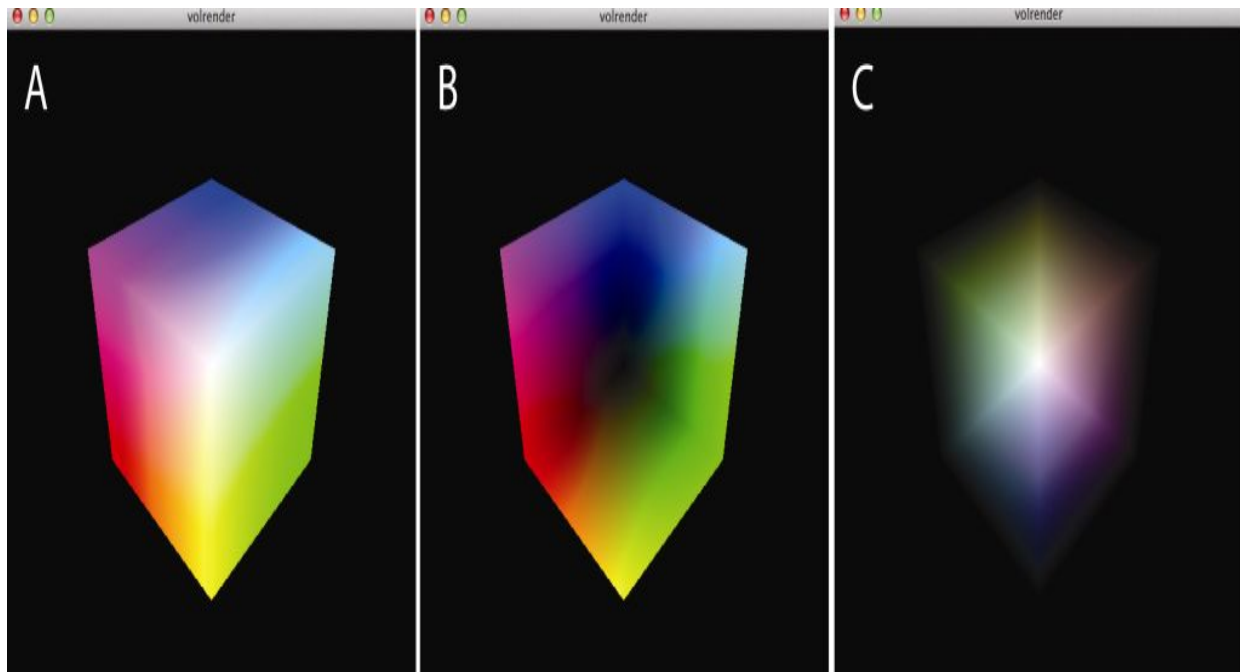


图11-4 渲染结果对比

## GPU渲染

在OpenGL中，渲染过程是通过FBO（Frame Buffer Object）来完成的。FBO是一个在GPU上创建的对象，它用于存储渲染结果。在渲染过程中，GPU会将渲染结果写入FBO，然后将其显示在屏幕上。FBO可以用于多种用途，例如：实现透明效果、实现后期处理效果、实现多视图渲染等。

在OpenGL中，FBO的创建和使用如下：

1. 创建FBO：在OpenGL中，FBO的创建是通过调用glGenFrameBuffers()函数来完成的。该函数会返回一个FBO的ID，该ID可以用于后续的渲染操作。

2. 绑定FBO：在渲染过程中，需要将FBO绑定到当前的渲染目标。这可以通过调用glBindFramebuffer()函数来实现。该函数会指定要绑定的FBO的ID。

3. 渲染：一旦FBO被绑定，就可以开始渲染了。渲染过程中，GPU会将渲染结果写入FBO。渲染完成后，可以将FBO绑定回默认的渲染目标（通常是屏幕），以便显示渲染结果。



makedata.py 生成测试数据

raycast.py 实现 RayCastRender 渲染器

raycube.py 实现 RayCastRender 渲染器  
RayCube 类

slicerender.py 实现 SliceRender 切片渲染器

volreader.py 实现基于 OpenGL 的  
体积数据加载器

volrender.py 实现基于 GLFW 的  
体积数据渲染器

运行 volrender.py 需要运行 makedata.py  
生成测试数据

[https://github.com/electronut/pp/tree/master/volrender/](https://github.com/electronut/pp/tree/master/volrender) 实现 glutils.py

<https://github.com/electronut/pp/tree/master/common/>

## 11.4 进阶

运行 volrender.py 需要运行 11.5

volreader.py 实现 11.5

```
def loadVolume(dirName):
```

```
 """read volume from directory as a 3D texture"""
```

```

 # list images in directory
❶ files = sorted(os.listdir(dirName))

 print('loading images from: %s' % dirName)

 imgDataList = []

 count = 0

 width, height = 0, 0

 for file in files:

❷
file_path = os.path.abspath(os.path.join(dirName, file))

 try:

 # read image

❸ img = Image.open(file_path)

 imgData = np.array(img.getdata(), np.uint8)

 # check if all images are of the same size

❹ if count is 0:

 width, height = img.size[0], img.size[1]

 imgDataList.append(imgData)

 else:

❺
if (width, height) == (img.size[0], img.size[1]):

 imgDataList.append(imgData)

 else:

 print('mismatch')

```

```

 raise RuntimeError("image size mismatch")

 count += 1

 #print img.size

except:

 # skip

 print('Invalid image: %s' % file_path)

load image data into single array

depth = count

❹ data = np.concatenate(imgDataList)

print('volume data dims: %d %d %d' % (width, height, depth)
)

load data into 3D texture

❺ texture = glGenTextures(1)

glPixelStorei(GL_UNPACK_ALIGNMENT, 1)

glBindTexture(GL_TEXTURE_3D, texture)

glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_CLAMP_
TO_EDGE)

glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_CLAMP_
TO_EDGE)

glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_CLAMP_

```

TO\_EDGE)

```
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
```

```
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
```

```
⑧ glTexImage3D(GL_TEXTURE_3D, 0, GL_RED,
 width, height, depth, 0,
 GL_RED, GL_UNSIGNED_BYTE, data)
```

```
return texture
```

```
⑨ return (texture, width, height, depth)
```

loadVolume() os.listdir() ① ②  
os.path.abspath() os.path.join() OS  
Python

③ Python PIL Image 8  
numpy

× ④ ⑤ numpy





```

list images in directory
files = sorted(os.listdir(dirName))
print('loading images from: %s' % dirName)
imgDataList = []
count = 0
width, height = 0, 0
for file in files:

file_path = os.path.abspath(os.path.join(dirName, file))
 try:
 # read image
 img = Image.open(file_path)
 imgData = np.array(img.getdata(), np.uint8)

 # check if all are of the same size
 if count is 0:
 width, height = img.size[0], img.size[1]
 imgDataList.append(imgData)
 else:

if (width, height) == (img.size[0], img.size[1]):
 imgDataList.append(imgData)
 else:

```

```

 print('mismatch')

 raise RuntimeError("image size mismatch")

 count += 1

 #print img.size

except:

 # skip

 print('Invalid image: %s' % file_path)

load image data into single array

depth = count

data = np.concatenate(imgDataList)

print('volume data dims: %d %d %d' % (width, height, depth)
)

load data into 3D texture

texture = glGenTextures(1)

glPixelStorei(GL_UNPACK_ALIGNMENT, 1)

glBindTexture(GL_TEXTURE_3D, texture)

glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_CLAMP_
TO_EDGE)

glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_CLAMP_
TO_EDGE)

```

```
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_CLAMP_
TO_EDGE)
```

```
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_MAG_FILTER, GL_LI
NEAR)
```

```
glTexParameterf(GL_TEXTURE_3D, GL_TEXTURE_MIN_FILTER, GL_LI
NEAR)
```

```
 glTexImage3D(GL_TEXTURE_3D, 0, GL_RED,
 width, height, depth, 0,
 GL_RED, GL_UNSIGNED_BYTE, data)
```

```
#return texture
```

```
 return (texture, width, height, depth)
```

```
load texture
```

```
def loadTexture(filename):
```

```
 img = Image.open(filename)
```

```
 img_data = np.array(list(img.getdata()), 'B')
```

```
 texture = glGenTextures(1)
```

```
 glPixelStorei(GL_UNPACK_ALIGNMENT,1)
```

```
 glBindTexture(GL_TEXTURE_2D, texture)
```

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_
TO_EDGE)
```

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_
```

```
TO_EDGE)
```

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
```

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
```

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, img.size[0], img.size[1],
```

```
0, GL_RGBA, GL_UNSIGNED_BYTE, img_data)
```

```
return texture
```

## 11.6 11.6

RayCube FBO raycube.py 11.7

```
❶ strVS = ""
```

```
#version 330 core
```

```
layout(location = 1) in vec3 cubePos;
```

```
layout(location = 2) in vec3 cubeCol;
```

```
uniform mat4 uMVMatrix;
```

```
uniform mat4 uPMatrix;
```

```
out vec4 vColor;
```

```
void main()
```

```
{
```

```
 // set back-face color
```

```
 vColor = vec4(cubeCol.rgb, 1.0);
```

```
 // transformed position
```

```
 vec4 newPos = vec4(cubePos.xyz, 1.0);
```

```
 // set position
```

```
 gl_Position = uPMatrix * uMVMMatrix * newPos;
```

```
}
```

```
"""
```

```
❷ strFS = """
```

```
#version 330 core
```

```
in vec4 vColor;
```

```
out vec4 fragColor;
```

```
void main()
```

```

{
 fragColor = vColor;
}

"""

```

❶ RayCube 클래스를 정의합니다. cubePos, cubeCol, uniform, uMVMMatrix, pMatrix, vColor, vColor를 정의합니다.

❷ RayCube 클래스를 정의합니다. vColor를 정의합니다.

## 11.6.1 RayCube 클래스

RayCube 클래스를 정의합니다.

```

cube vertices
❶ vertices = numpy.array([
 0.0, 0.0, 0.0,
 1.0, 0.0, 0.0,
 1.0, 1.0, 0.0,
 0.0, 1.0, 0.0,
 0.0, 0.0, 1.0,
 1.0, 0.0, 1.0,
 1.0, 1.0, 1.0,
 0.0, 1.0, 1.0

```

```
], numpy.float32)
```

```
cube colors
```

```
❷ colors = numpy.array([
 0.0, 0.0, 0.0,
 1.0, 0.0, 0.0,
 1.0, 1.0, 0.0,
 0.0, 1.0, 0.0,
 0.0, 0.0, 1.0,
 1.0, 0.0, 1.0,
 1.0, 1.0, 1.0,
 0.0, 1.0, 1.0
], numpy.float32)
```

```
individual triangles
```

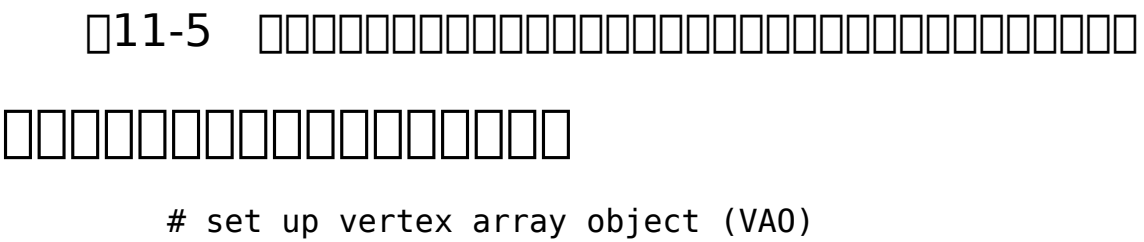
```
❸ indices = numpy.array([
 4, 5, 7,
 7, 5, 6,
 5, 1, 6,
 6, 1, 2,
 1, 0, 2,
 2, 0, 3,
 0, 4, 3,
```

```
3, 4, 7,
6, 2, 7,
7, 2, 3,
4, 0, 5,
5, 0, 1
], numpy.int16)
```

RayCube 프로그램  
① ②

6 6×6 36  
36 8  
indices ③ 11-5





```

self.vao = glGenVertexArrays(1)

glBindVertexArray(self.vao)

vertex buffer

self.vertexBuffer = glGenBuffers(1)

glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(vertices), vertices, GL
_STATIC_DRAW)

vertex buffer – cube vertex colors

self.colorBuffer = glGenBuffers(1)

glBindBuffer(GL_ARRAY_BUFFER, self.colorBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(colors), colors, GL_STA
TIC_DRAW)

index buffer

self.indexBuffer = glGenBuffers(1)

❶
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, self.indexBuffer);

glBufferData(GL_ELEMENT_ARRAY_BUFFER, 2*len(indices), indic
es,

GL_STATIC_DRAW)

```

VertexArrayObject (VAO) 객체를 생성하고 indices 배열을 GL\_ELEMENT\_ARRAY\_BUFFER로 등록하여 GPU에 업로드한다.

## 11.6.2 VAO 사용하기

다음은 VAO를 사용하여 프레임 버퍼와 텍스처를 설정하는 코드이다.

```
def initFB0(self):
 # create frame buffer object
 self.fboHandle = glGenFramebuffers(1)

 # create texture
 self.texHandle = glGenTextures(1)

 # create depth buffer
 self.depthHandle = glGenRenderbuffers(1)

 # bind
 glBindFramebuffer(GL_FRAMEBUFFER, self.fboHandle)

 glActiveTexture(GL_TEXTURE0)
 glBindTexture(GL_TEXTURE_2D, self.texHandle)

 # set parameters to draw the image at different sizes
```

①  
glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

NEAR)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE)
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE)
```

```
set up texture
```

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, self.width, self.height,
```

```
0, GL_RGBA, GL_UNSIGNED_BYTE, None)
```

```
bind texture to FB0
```

```
② glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
,
GL_TEXTURE_2D, self.texHandle, 0)
```

```
bind
```

```
③ glBindRenderbuffer(GL_RENDERBUFFER, self.depthHandle)
```

```
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT24,
,
self.width, self.height)
```

```

 # bind depth buffer to FBO

glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT,

GL_RENDERBUFFER, self.depthHandle)

 # check status

 ❷ status = glCheckFramebufferStatus(GL_FRAMEBUFFER)

 if status == GL_FRAMEBUFFER_COMPLETE:

 pass

 #print "fbo %d complete" % self.fboHandle

 elif status == GL_FRAMEBUFFER_UNSUPPORTED:

 print "fbo %d unsupported" % self.fboHandle

 else:

 print "fbo %d Error" % self.fboHandle

```

```

 ❶
 ❷
 ❸
 ❹
 24

```

## 11.6.3 渲染背面

```

def renderBackFace(self, pMatrix, mvMatrix):

```

```

 def renderBackFace(self, pMatrix, mvMatrix):

```

```
 """renders back-face of ray-
cube to a texture and returns it"""
```

```
 # render to FBO
```

```
❶ glBindFramebuffer(GL_FRAMEBUFFER, self.fboHandle)
```

```
 # set active texture
```

```
glActiveTexture(GL_TEXTURE0)
```

```
 # bind to FBO texture
```

```
glBindTexture(GL_TEXTURE_2D, self.texHandle)
```

```
 # render cube with face culling enabled
```

```
❷ self.renderCube(pMatrix, mvMatrix, self.program, True)
```

```
 # unbind texture
```

```
❸ glBindTexture(GL_TEXTURE_2D, 0)
```

```
glBindFramebuffer(GL_FRAMEBUFFER, 0)
```

```
glBindRenderbuffer(GL_RENDERBUFFER, 0)
```

```
 # return texture ID
```

```
❹ return self.texHandle
```

```
❶ glBindFramebuffer(GL_FRAMEBUFFER, self.fboHandle)
❷ self.renderCube(pMatrix, mvMatrix, self.program, True)
❸ glBindTexture(GL_TEXTURE_2D, 0)
❹ return self.texHandle
```

FBO를 생성하고, FBO를 ID로  
인식할 수 있다

## 11.6.4 FBO를 사용하여 렌더링

11.6.3에서 renderCube() 함수를 호출할 때  
False를 전달한다

```
def renderFrontFace(self, pMatrix, mvMatrix, program):
 """render front-face of ray-cube"""

 # no face culling

 self.renderCube(pMatrix, mvMatrix, program, False)
```

## 11.6.5 FBO를 사용하여 렌더링

renderCube() 함수를 호출할 때

```
def renderCube(self, pMatrix, mvMatrix, program, cullFace):
 """renderCube uses face culling if flag set"""

 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

 # set shader program
 glUseProgram(program)

 # set projection matrix
```

```
glUniformMatrix4fv(glGetUniformLocation(program, b'uPMatrix'),
```

```
1, GL_FALSE, pMatrix)
```

```
set modelview matrix
```

```
glUniformMatrix4fv(glGetUniformLocation(program, b'uMVMatrix'),
```

```
1, GL_FALSE, mvMatrix)
```

```
enable face culling
```

```
glDisable(GL_CULL_FACE)
```

```
❶ if cullFace:
```

```
 glFrontFace(GL_CCW)
```

```
 glCullFace(GL_FRONT)
```

```
 glEnable(GL_CULL_FACE)
```

```
bind VAO
```

```
glBindVertexArray(self.vao)
```

```
animated slice
```

```
❷
```

```
glDrawElements(GL_TRIANGLES, self.nIndices, GL_UNSIGNED_SHORT,
```



RT, None)

```
unbind VA0

glBindVertexArray(0)

reset cull face

if cullFace:

 # disable face culling

 glDisable(GL_CULL_FACE)
```

```
#####
❶
#####glDrawElements()❷#####
#####
```

## 11.6.6 窗口大小调整

FBO窗口大小调整时，需要重新创建FBO，并重新加载RayCube数据。

```
❶ def reshape(self, width, height):

 self.width = width

 self.height = height

 self.aspect = width/float(height)

 # re-create FBO
```

```
self.clearFB0()
```

```
self.initFB0()
```

OpenGLreshape()❶

## 11.7

<https://github.com/electronut/pp/tree/master/volrender/raycube.py>

```
import OpenGL
```

```
from OpenGL.GL import *
```

```
from OpenGL.GL.shaders import *
```

```
import numpy, math, sys
```

```
import volreader, glutils
```

```
strVS = """
```

```
#version 330 core
```

```
layout(location = 1) in vec3 cubePos;
```

```
layout(location = 2) in vec3 cubeCol;
```

```
uniform mat4 uMVMatrix;
```

```
uniform mat4 uPMatrix;
```

```
out vec4 vColor;
```

```

void main()
{
 // set back face color
 vColor = vec4(cubeCol.rgb, 1.0);

 // transformed position
 vec4 newPos = vec4(cubePos.xyz, 1.0);

 // set position
 gl_Position = uPMatrix * uMVMMatrix * newPos;

}

"""

strFS = """
#version 330 core

in vec4 vColor;
out vec4 fragColor;

void main()
{

```

```
 fragColor = vColor;
 }
 """
```

```
class RayCube:
```

```
 """class used to generate rays used in ray casting"""
```

```
 def __init__(self, width, height):
```

```
 """RayCube constructor"""
```

```
 # set dims
```

```
 self.width, self.height = width, height
```

```
 # create shader
```

```
 self.program = glutils.loadShaders(strVS, strFS)
```

```
 # cube vertices
```

```
 vertices = numpy.array([
```

```
 0.0, 0.0, 0.0,
```

```
 1.0, 0.0, 0.0,
```

```
 1.0, 1.0, 0.0,
```

```
 0.0, 1.0, 0.0,
```

```
 0.0, 0.0, 1.0,
```

```
1.0, 0.0, 1.0,
1.0, 1.0, 1.0,
0.0, 1.0, 1.0
], numpy.float32)
```

```
cube colors
```

```
colors = numpy.array([
 0.0, 0.0, 0.0,
 1.0, 0.0, 0.0,
 1.0, 1.0, 0.0,
 0.0, 1.0, 0.0,
 0.0, 0.0, 1.0,
 1.0, 0.0, 1.0,
 1.0, 1.0, 1.0,
 0.0, 1.0, 1.0
], numpy.float32)
```

```
individual triangles
```

```
indices = numpy.array([
 4, 5, 7,
 7, 5, 6,
 5, 1, 6,
```

```
 6, 1, 2,
 1, 0, 2,
 2, 0, 3,
 0, 4, 3,
 3, 4, 7,
 6, 2, 7,
 7, 2, 3,
 4, 0, 5,
 5, 0, 1
], numpy.int16)
```

```
self.nIndices = indices.size
```

```
set up vertex array object (VAO)
```

```
self.vao = glGenVertexArrays(1)
```

```
glBindVertexArray(self.vao)
```

```
#vertex buffer
```

```
self.vertexBuffer = glGenBuffers(1)
```

```
glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)
```

```
glBufferData(GL_ARRAY_BUFFER, 4*len(vertices), vertices, GL
_STATIC_DRAW)
```

```

 # vertex buffer - cube vertex colors

 self.colorBuffer = glGenBuffers(1)

 glBindBuffer(GL_ARRAY_BUFFER, self.colorBuffer)

 glBufferData(GL_ARRAY_BUFFER, 4*len(colors), colors, GL_STATIC_DRAW);

 # index buffer

 self.indexBuffer = glGenBuffers(1)

 glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, self.indexBuffer);

 glBufferData(GL_ELEMENT_ARRAY_BUFFER, 2*len(indices), indices,

 GL_STATIC_DRAW)

 # enable attrs using the layout indices in shader

 aPosLoc = 1
 aColorLoc = 2

 # bind buffers:

 glEnableVertexAttribArray(1)
 glEnableVertexAttribArray(2)

 # vertex

```

```

 glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glVertexAttribPointer(aPosLoc, 3, GL_FLOAT, GL_FALSE, 0, None)

 # color

 glBindBuffer(GL_ARRAY_BUFFER, self.colorBuffer)

glVertexAttribPointer(aColorLoc, 3, GL_FLOAT, GL_FALSE, 0,
None)

 # index

glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, self.indexBuffer)

 # unbind VAO

glBindVertexArray(0)

 # FBO

self.initFBO()

def renderBackFace(self, pMatrix, mvMatrix):
 """renders back-face of ray-
cube to a texture and returns it"""

 # render to FBO

 glBindFramebuffer(GL_FRAMEBUFFER, self.fboHandle)

```



```

 # set active texture
 glActiveTexture(GL_TEXTURE0)

 # bind to FBO texture
 glBindTexture(GL_TEXTURE_2D, self.texHandle)

 # render cube with face culling enabled

self.renderCube(pMatrix, mvMatrix, self.program, True)

 # unbind texture
 glBindTexture(GL_TEXTURE_2D, 0)
 glBindFramebuffer(GL_FRAMEBUFFER, 0)
 glBindRenderbuffer(GL_RENDERBUFFER, 0)

 # return texture ID
 return self.texHandle

def renderFrontFace(self, pMatrix, mvMatrix, program):
 """render front face of ray-cube"""
 # no face culling
 self.renderCube(pMatrix, mvMatrix, program, False)

def renderCube(self, pMatrix, mvMatrix, program, cullFace):

```

```

"""render cube use face culling if flag set"""

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

set shader program
glUseProgram(program)

set projection matrix

glUniformMatrix4fv(glGetUniformLocation(program, b'uPMatrix
'),
 1, GL_FALSE, pMatrix)

set modelview matrix

glUniformMatrix4fv(glGetUniformLocation(program, b'uMVMatri
x'),
 1, GL_FALSE, mvMatrix)

enable face culling
glDisable(GL_CULL_FACE)
if cullFace:
 glFrontFace(GL_CCW)
 glCullFace(GL_FRONT)
 glEnable(GL_CULL_FACE)

```

```

bind VAO

glBindVertexArray(self.vao)

animated slice

glDrawElements(GL_TRIANGLES, self.nIndices, GL_UNSIGNED_SHORT, None)

unbind VAO

glBindVertexArray(0)

reset cull face

if cullFace:
 # disable face culling
 glDisable(GL_CULL_FACE)

def reshape(self, width, height):
 self.width = width
 self.height = height
 self.aspect = width/float(height)
 # re-create FBO
 self.clearFBO()

```

```

 self.initFB0()

def initFB0(self):
 # create frame buffer object
 self.fboHandle = glGenFramebuffers(1)

 # create texture
 self.texHandle = glGenTextures(1)

 # create depth buffer
 self.depthHandle = glGenRenderbuffers(1)

 # bind
 glBindFramebuffer(GL_FRAMEBUFFER, self.fboHandle)

 glActiveTexture(GL_TEXTURE0)
 glBindTexture(GL_TEXTURE_2D, self.texHandle)

 # set parameters to draw the image at different sizes

 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)

 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)

 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_

```

```
TO_EDGE)
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_
TO_EDGE)
```

```
set up texture
```

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, self.width, self.he
ight,
```

```
0, GL_RGBA, GL_UNSIGNED_BYTE, None)
```

```
bind texture to FB0
```

```
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0
,
```

```
GL_TEXTURE_2D, self.texHandle, 0)
```

```
bind
```

```
glBindRenderbuffer(GL_RENDERBUFFER, self.depthHandle)
```

```
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT24
,
```

```
self.width, self.height)
```

```
bind depth buffer to FB0
```

```
glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHME
NT,
```

```
GL_RENDERBUFFER, self.depthHandle)
```

```
 # check status
```

```
 status = glCheckFramebufferStatus(GL_FRAMEBUFFER)
```

```
 if status == GL_FRAMEBUFFER_COMPLETE:
```

```
 pass
```

```
 #print "fbo %d complete" % self.fboHandle
```

```
 elif status == GL_FRAMEBUFFER_UNSUPPORTED:
```

```
 print("fbo %d unsupported" % self.fboHandle)
```

```
 else:
```

```
 print("fbo %d Error" % self.fboHandle)
```

```
glBindTexture(GL_TEXTURE_2D, 0)
```

```
glBindFramebuffer(GL_FRAMEBUFFER, 0)
```

```
glBindRenderbuffer(GL_RENDERBUFFER, 0)
```

```
return
```

```
def clearFB0(self):
```

```
 """clears old FB0"""
```

```
 # delete FB0
```

```
 if glIsFramebuffer(self.fboHandle):
```

```

 glDeleteFramebuffers(int(self.fboHandle))

delete texture
if glIsTexture(self.texHandle):
 glDeleteTextures(int(self.texHandle))

def close(self):
 """call this to free up OpenGL resources"""
 glBindTexture(GL_TEXTURE_2D, 0)
 glBindFramebuffer(GL_FRAMEBUFFER, 0)
 glBindRenderbuffer(GL_RENDERBUFFER, 0)
 # delete FBO
 if glIsFramebuffer(self.fboHandle):
 glDeleteFramebuffers(int(self.fboHandle))

delete texture
if glIsTexture(self.texHandle):
 glDeleteTextures(int(self.texHandle))

delete render buffer
"""

```

```

if glIsRenderbuffer(self.depthHandle):
 glDeleteRenderbuffers(1, int(self.depthHandle))

 """

delete buffers
"""

glDeleteBuffers(1, self._vertexBuffer)

glDeleteBuffers(1, &_amp;_indexBuffer)

glDeleteBuffers(1, &_amp;_colorBuffer)

"""

```

## 11.8 RayCastRender

RayCastRenderは、RayCubeを生成して、raycast.pyで11.9で

RayCubeを生成して、raycast.pyで11.9で

```

def __init__(self, width, height, volume):
 """RayCastRender construction"""

```

```

create RayCube object

```

```

❶ self.raycube = raycube.RayCube(width, height)

```

```

set dimensions

```

```

self.width = width

```



```

self.height = height

self.aspect = width/float(height)

create shader

❷ self.program = glutils.loadShaders(strVS, strFS)

texture

❸ self.texVolume, self.Nx, self.Ny, self.Nz = volume

initialize camera

❹ self.camera = Camera()

```

❶ RayCube ❷  
 ❸ OpenGL  
 RayCastRender ❹  
 Camera OpenGL  
 10

RayCastRender

```

def draw(self):

 # build projection matrix

 ❶ pMatrix = glutils.perspective(45.0, self.aspect, 0.1, 100.0
)

 # modelview matrix

 ❷ mvMatrix = glutils.lookAt(self.camera.eye, self.camera.cent

```

er,

self.camera.up)

# render

# generate ray-cube back-face texture

③

texture = self.raycube.renderBackFace(pMatrix, mvMatrix)

# set shader program

④

glUseProgram(self.program)

# set window dimensions

glUniform2f(glGetUniformLocation(self.program, b"uWinDims"),  
,

float(self.width), float(self.height))

# bind to texture unit 0, which represents back-  
faces of cube

⑤

glActiveTexture(GL\_TEXTURE0)

glBindTexture(GL\_TEXTURE\_2D, texture)

glUniform1i(glGetUniformLocation(self.program, b"textBackFaces"), 0)

```

 # texture unit 1: 3D volume texture

❹ glActiveTexture(GL_TEXTURE1)

 glBindTexture(GL_TEXTURE_3D, self.texVolume)

glUniform1i(glGetUniformLocation(self.program, b"texVolume"
), 1)

 # draw front-face of cubes

❺ self.raycube.renderFrontFace(pMatrix, mvMatrix, self.progra
m)

```

❶ `glutils.perspective()` 设置透视投影  
 ❷ `glutils.lookAt()` 设置观察点  
 ❸ `RayCube.renderBackFace()` 渲染背面  
 设置模型ID

❹ 设置模型ID ❺ 设置模型ID ❸ 设置模型ID  
 0 ❻ 设置模型ID  
 1 设置模型ID  
 ❷ `RayCube.renderFrontFace()` 渲染正面  
`RayCastRender` 渲染射线

## 11.8.1 射线追踪

`RayCastRender` 渲染射线

```
#version 330 core
```

```
❶ layout(location = 1) in vec3 cubePos;
```

```
 layout(location = 2) in vec3 cubeCol;
```

```
❷ uniform mat4 uMVMatrix;
```

```
 uniform mat4 uPMatrix;
```

```
❸ out vec4 vColor;
```

```
void main()
```

```
{
```

```
 // set position
```

```
❹ gl_Position = uPMatrix * uMVMatrix * vec4(cubePos.xyz, 1.0);
```

```
 // set color
```

```
❺ vColor = vec4(cubeCol.rgb, 1.0);
```

```
}
```

❶ RayCube  
RayCastRender VBO  
❷

③ ④ ⑤  
gl\_Position  
⑤

## 11.8.2

```
#version 330 core
```

```
in vec4 vColor;
```

```
uniform sampler2D texBackFaces;
```

```
uniform sampler3D texVolume;
```

```
uniform vec2 uWinDims;
```

```
out vec4 fragColor;
```

```
void main()
```

```
{
```

```
 // start of ray
```

```
 ❶ vec3 start = vColor.rgb;
```

```
 // calculate texture coordinates at fragment,
```

```
 // which is a fraction of window coordinates
```

```

❷ vec2 texc = gl_FragCoord.xy/uWinDims.xy;

 // get end of ray by looking up back-face color
❸ vec3 end = texture(texBackFaces, texc).rgb;

 // calculate ray direction
❹ vec3 dir = end - start;

 // normalized ray direction
 vec3 norm_dir = normalize(dir);

 // the length from front to back is calculated and
 // used to terminate the ray
 float len = length(dir.xyz);

 // ray step size
 float stepSize = 0.01;

 // x-ray projection
 vec4 dst = vec4(0.0);

 // step through the ray

```

```

❶ for(float t = 0.0; t < len; t += stepSize) {

 // set position to end point of ray

❷ vec3 samplePos = start + t*norm_dir;

 // get texture value at position

❸ float val = texture(texVolume, samplePos).r;
 vec4 src = vec4(val);

 // set opacity

❹ src.a *= 0.1;
 src.rgb *= src.a;

 // blend with previous value

❺ dst = (1.0 - dst.a)*src + dst;

 // exit loop when alpha exceeds threshold

❻ if(dst.a >= 0.95)
 break;
}

// set fragment color
fragColor = dst;

```

}

编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`，  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` OpenGL 库。

编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` ① 编译选项为  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` 11.1.2 编译选项为  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`

② 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` [0,1] ③ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`

④ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 ⑤ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 ⑥ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` ⑦ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`

编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` ⑧ ⑨ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` dst 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` alpha 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` alpha 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`

⑩ 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` alpha 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` 0.95 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` “ ” x 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`  
编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT` alpha 编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`

## 11.9 编译选项

编译选项为 `-std=c++11 -DGL_GLEXT=GL_GLEXT`

<https://github.com/electronut/pp/tree/mast>



## er/volrender/ raycast.py

```
import OpenGL

from OpenGL.GL import *

from OpenGL.GL.shaders import *

import numpy as np

import math, sys

import raycube, glutils, volreader

strVS = """

#version 330 core

layout(location = 1) in vec3 cubePos;
layout(location = 2) in vec3 cubeCol;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;

out vec4 vColor;

void main()
{
```

```

 // set position

gl_Position = uPMatrix * uMVMMatrix * vec4(cubePos.xyz, 1.0)
;

 // set color

 vColor = vec4(cubeCol.rgb, 1.0);
 }
 ""

 strFS = ""
 #version 330 core

 in vec4 vColor;

 uniform sampler2D texBackFaces;
 uniform sampler3D texVolume;
 uniform vec2 uWinDims;

 out vec4 fragColor;

 void main()
 {
 // start of ray

```

```
vec3 start = vColor.rgb;

// calculate texture coords at fragment,
// which is a fraction of window coords
vec2 texc = gl_FragCoord.xy/uWinDims.xy;

// get end of ray by looking up back-face color
vec3 end = texture(texBackFaces, texc).rgb;

// calculate ray direction
vec3 dir = end - start;

// normalized ray direction
vec3 norm_dir = normalize(dir);

// the length from front to back is calculated and
// used to terminate the ray
float len = length(dir.xyz);

// ray step size
float stepSize = 0.01;

// x-ray projection
```

```

vec4 dst = vec4(0.0);

// step through the ray
for(float t = 0.0; t < len; t += stepSize) {

 // set position to end point of ray
 vec3 samplePos = start + t*norm_dir;

 // get texture value at position
 float val = texture(texVolume, samplePos).r;
 vec4 src = vec4(val);

 // set opacity
 src.a *= 0.1;
 src.rgb *= src.a;

 // blend with previous value
 dst = (1.0 - dst.a)*src + dst;

 // exit loop when alpha exceeds threshold
 if(dst.a >= 0.95)
 break;
}

```

```

 }

 // set fragment color
 fragColor = dst;
}

"""

class Camera:
 """helper class for viewing"""
 def __init__(self):
 self.r = 1.5
 self.theta = 0
 self.center = [0.5, 0.5, 0.5]
 self.eye = [0.5 + self.r, 0.5, 0.5]
 self.up = [0.0, 0.0, 1.0]

 def rotate(self, clockWise):
 """rotate eye by one step"""
 if clockWise:
 self.theta = (self.theta + 5) % 360
 else:
 self.theta = (self.theta - 5) % 360
 # recalculate eye

```

```
self.eye = [0.5 + self.r*math.cos(math.radians(self.theta))
,
0.5 + self.r*math.sin(math.radians(self.theta)),
0.5]
```

```
class RayCastRender:
```

```
 """class that does Ray Casting"""
```

```
 def __init__(self, width, height, volume):
```

```
 """RayCastRender constr"""
```

```
 # create RayCube object
```

```
 self.raycube = raycube.RayCube(width, height)
```

```
 # set dimensions
```

```
 self.width = width
```

```
 self.height = height
```

```
 self.aspect = width/float(height)
```

```
 # create shader
```

```
 self.program = glutils.loadShaders(strVS, strFS)
```

```
 # texture
```

```
 self.texVolume, self.Nx, self.Ny, self.Nz = volume
```

```
initialize camera

self.camera = Camera()

def draw(self):

 # build projection matrix

 pMatrix = glutils.perspective(45.0, self.aspect, 0.1, 100.0
)

 # modelview matrix

 mvMatrix = glutils.lookAt(self.camera.eye, self.camera.center,
 self.camera.up)

 # render

 # generate ray-cube back-face texture

 texture = self.raycube.renderBackFace(pMatrix, mvMatrix)

 # set shader program

 glUseProgram(self.program)
```

```

 # set window dimensions

glUniform2f(glGetUniformLocation(self.program, b"uWinDims")
,
 float(self.width), float(self.height))

texture unit 0, which represents back-faces of cube
glActiveTexture(GL_TEXTURE0)
glBindTexture(GL_TEXTURE_2D, texture)

glUniform1i(glGetUniformLocation(self.program, b"texBackFaces"), 0)

texture unit 1: 3D volume texture
glActiveTexture(GL_TEXTURE1)
glBindTexture(GL_TEXTURE_3D, self.texVolume)

glUniform1i(glGetUniformLocation(self.program, b"texVolume"), 1)

draw front face of cubes

self.raycube.renderFrontFace(pMatrix, mvMatrix, self.program)

#self.render(pMatrix, mvMatrix)

```



```

def keyPressed(self, key):
 if key == 'l':
 self.camera.rotate(True)
 elif key == 'r':
 self.camera.rotate(False)

def reshape(self, width, height):
 self.width = width
 self.height = height
 self.aspect = width/float(height)
 self.raycube.reshape(width, height)

def close(self):
 self.raycube.close()

```

## 11.10 练习

在本节中，我们将实现一个名为 `SliceRender` 的类，用于渲染切片。  
 该类的实现文件名为 `slicerender.py`，位于第 11.11 节。

在本节中，我们将实现一个名为 `SliceRender` 的类，用于渲染切片。

```

set up vertex array object (VAO)

self.vao = glGenVertexArrays(1)

```

```

glBindVertexArray(self.vao)

define quad vertices
❶ vertexData = numpy.array([0.0, 1.0, 0.0,
 0.0, 0.0, 0.0,
 1.0, 1.0, 0.0,
 1.0, 0.0, 0.0], numpy.float32)

vertex buffer
self.vertexBuffer = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData
,

 GL_STATIC_DRAW)

enable arrays
glEnableVertexAttribArray(self.vertIndex)

set buffers
glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glVertexAttribPointer(self.vertIndex, 3, GL_FLOAT, GL_FALSE
, 0, None)

unbind VAO
glBindVertexArray(0)

```

我们使用VAO和VBO来管理顶点数据①  
我们使用xy来指定顶点位置  
GL\_TRIANGLE\_STRIP来指定三角形  
我们使用xyz来指定顶点位置  
我们使用x来指定顶点位置

我们使用SliceRender来管理切片

```
def draw(self):
 # clear buffers
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
 # build projection matrix
 ①
 pMatrix = glutils.ortho(-0.6, 0.6, -0.6, 0.6, 0.1, 100.0)
 # modelview matrix
 ②
 mvMatrix = numpy.array([1.0, 0.0, 0.0, 0.0,
 0.0, 1.0, 0.0, 0.0,
 0.0, 0.0, 1.0, 0.0,
 -0.5, -0.5, -1.0, 1.0], numpy.float32)
 # use shader
 glUseProgram(self.program)
 # set projection matrix
 glUniformMatrix4fv(self.pMatrixUniform, 1, GL_FALSE, pMatrix)
```

```

 # set modelview matrix

glUniformMatrix4fv(self.mvMatrixUniform, 1, GL_FALSE, mvMatrix)

 # set current slice fraction

③
glUniform1f(glGetUniformLocation(self.program, b"uSliceFrac"),

float(self.currSliceIndex)/float(self.currSliceMax))

 # set current slice mode

④
glUniform1i(glGetUniformLocation(self.program, b"uSliceMode"),

 self.mode)

 # enable texture

glActiveTexture(GL_TEXTURE0)

glBindTexture(GL_TEXTURE_3D, self.texture)

glUniform1i(glGetUniformLocation(self.program, b"text"), 0)

 # bind VAO

glBindVertexArray(self.vao)

```

```
draw

glDrawArrays(GL_TRIANGLE_STRIP, 0, 4)

unbind VAO

glBindVertexArray(0)
```

1. `glOrtho()` の引数として、`0.1` を指定する。  
 2. `glOrtho()` の引数として、`0` を指定する。  
 3. `glOrtho()` の引数として、`-0.5` を指定する。  
 4. `glOrtho()` の引数として、`10` を指定する。  
 x, y, z の範囲を `0` から `1` まで指定する。

## 11.10.1 シェーダ

シェーダ `SliceRender` のソースコード

```
version 330 core

in vec3 aVert;

uniform mat4 uMVMatrix;

uniform mat4 uPMatrix;

uniform float uSliceFrac;
```

```

uniform int uSliceMode;

out vec3 texcoord;

void main() {

 // x slice
 if (uSliceMode == 0) {
 ❶ texcoord = vec3(uSliceFrac, aVert.x, 1.0-aVert.y);
 }
 // y slice
 else if (uSliceMode == 1) {
 ❷ texcoord = vec3(aVert.x, uSliceFrac, 1.0-aVert.y);
 }
 // z slice
 else {
 ❸ texcoord = vec3(aVert.x, 1.0-aVert.y, uSliceFrac);
 }

 // calculate transformed vertex
 gl_Position = uPMatrix * uMVMMatrix * vec4(aVert, 1.0);
}

```

uniform texture3D tex;
uniform float f;
uniform vec3 Vx, Vy;

1. texture3D(tex, vec3(x, y, z));
yz texture3D(tex, vec3(x, y, z));
1] texture3D(tex, vec3(x, y, z));
f, Vx, Vy texture3D(tex, vec3(x, y, z));
Vx, Vy texture3D(tex, vec3(x, y, z));
OpenGL texture3D(tex, vec3(x, y, z));
t texture3D(tex, vec3(x, y, z));
1 - t texture3D(tex, vec3(x, y, z));
Vy 1 texture3D(tex, vec3(x, y, z));
2 texture3D(tex, vec3(x, y, z));
3 texture3D(tex, vec3(x, y, z));
yz texture3D(tex, vec3(x, y, z));

## 11.10.2 Shader Code

Shader Code

```
version 330 core
```

```
1 in vec3 texcoord;
```

```
2 uniform sampler3D texture;
```

```
out vec4 fragColor;
```

```
void main() {
```

```
 // look up color in texture
```

```

❸ vec4 col = texture(tex, texcoord);

❹ fragColor = col.rrra;

}

```

❶ texcoord 纹理坐标  
 ❷ uniform 纹理坐标  
 ❸ texcoord 纹理坐标  
 ❹ fragColor 纹理坐标  
 col.rrra

### 11.10.3 切片渲染

切片渲染器 SliceRender

```

def keyPressed(self, key):
 """keypress handler"""

 if key == 'x':
❶ self.mode = SliceRender.XSLICE
 # reset slice index
 self.currSliceIndex = int(self.Nx/2)
 self.currSliceMax = self.Nx

 elif key == 'y':
 self.mode = SliceRender.YSLICE
 # reset slice index
 self.currSliceIndex = int(self.Ny/2)
 self.currSliceMax = self.Ny

```



```

elif key == 'z':

 self.mode = SliceRender.ZSLICE

 # reset slice index

 self.currSliceIndex = int(self.Nz/2)

 self.currSliceMax = self.Nz

elif key == 'l':

```

```

❷
self.currSliceIndex = (self.currSliceIndex + 1) % self.curr
SliceMax

```

```

elif key == 'r':

```

```

self.currSliceIndex = (self.currSliceIndex - 1) % self.curr
SliceMax

```

1. 2.
   
 SliceRender
   
 x y z
   
 1. 2.
   
 %
   
 " " 0

## 11.11

<https://github.com/electronut/pp/tree/master/volrender/pp slicerender.py>

```

import OpenGL

from OpenGL.GL import *

```

```
from OpenGL.GL.shaders import *
import numpy, math, sys

import volreader, glutils

strVS = """
version 330 core

in vec3 aVert;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;

uniform float uSliceFrac;
uniform int uSliceMode;

out vec3 texcoord;

void main() {

 // x slice
 if (uSliceMode == 0) {
 texcoord = vec3(uSliceFrac, aVert.x, 1.0-aVert.y);
```

```

 }

 // y slice
 else if (uSliceMode == 1) {
 texcoord = vec3(aVert.x, uSliceFrac, 1.0-aVert.y);
 }

 // z slice
 else {
 texcoord = vec3(aVert.x, 1.0-aVert.y, uSliceFrac);
 }

 // calculate transformed vertex
 gl_Position = uPMatrix * uMVMatrix * vec4(aVert, 1.0);
}

"""

strFS = """

version 330 core

in vec3 texcoord;

uniform sampler3D tex;

out vec4 fragColor;

```

```

void main() {
 // look up color in texture
 vec4 col = texture(tex, texcoord);
 fragColor = col.rrra;
}

```

```

"""

```

```

class SliceRender:
 # slice modes
 XSLICE, YSLICE, ZSLICE = 0, 1, 2

 def __init__(self, width, height, volume):
 """SliceRender constructor"""
 self.width = width
 self.height = height
 self.aspect = width/float(height)

 # slice mode
 self.mode = SliceRender.ZSLICE

```

```

 # create shader

self.program = glutils.loadShaders(strVS, strFS)

 glUseProgram(self.program)

self.pMatrixUniform = glGetUniformLocation(self.program, b
'uPMatrix')

self.mvMatrixUniform = glGetUniformLocation(self.program,

b"uMVMatrix")

 # attributes

self.vertIndex = glGetAttribLocation(self.program, b"aVert
")

 # set up vertex array object (VAO)

self.vao = glGenVertexArrays(1)

glBindVertexArray(self.vao)

 # define quad vertices

vertexData = numpy.array([0.0, 1.0, 0.0,

 0.0, 0.0, 0.0,

 1.0, 1.0, 0.0,

```

```

1.0, 0.0, 0.0], numpy.float32)

 # vertex buffer

 self.vertexBuffer = glGenBuffers(1)

glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glBufferData(GL_ARRAY_BUFFER, 4*len(vertexData), vertexData,

GL_STATIC_DRAW)

 # enable arrays

glEnableVertexAttribArray(self.vertIndex)

 # set buffers

glBindBuffer(GL_ARRAY_BUFFER, self.vertexBuffer)

glVertexAttribPointer(self.vertIndex, 3, GL_FLOAT, GL_FALSE, 0, None)

 # unbind VAO

glBindVertexArray(0)

load texture

self.texture, self.Nx, self.Ny, self.Nz = volume

current slice index

self.currSliceIndex = int(self.Nz/2);

```

```

 self.currSliceMax = self.Nz;

def reshape(self, width, height):
 self.width = width
 self.height = height
 self.aspect = width/float(height)

def draw(self):
 # clear buffers
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
 # build projection matrix

pMatrix = glutils.ortho(-0.6, 0.6, -0.6, 0.6, 0.1, 100.0)
 # modelview matrix
 mvMatrix = numpy.array([1.0, 0.0, 0.0, 0.0,
 0.0, 1.0, 0.0, 0.0,
 0.0, 0.0, 1.0, 0.0,
 -0.5, -0.5, -1.0, 1.0], numpy.flo
at32)

 # use shader
 glUseProgram(self.program)

```

```

 # set projection matrix

glUniformMatrix4fv(self.pMatrixUniform, 1, GL_FALSE, pMatrix)

 # set modelview matrix

glUniformMatrix4fv(self.mvMatrixUniform, 1, GL_FALSE, mvMatrix)

 # set current slice fraction

glUniform1f(glGetUniformLocation(self.program, b"uSliceFraction"),

float(self.currSliceIndex)/float(self.currSliceMax))

 # set current slice mode

glUniform1i(glGetUniformLocation(self.program, b"uSliceMode"),

 self.mode)

 # enable texture

glActiveTexture(GL_TEXTURE0)

glBindTexture(GL_TEXTURE_3D, self.texture)

glUniform1i(glGetUniformLocation(self.program, b"tex"), 0)

```



```

bind VAO

glBindVertexArray(self.vao)

draw

glDrawArrays(GL_TRIANGLE_STRIP, 0, 4)

unbind VAO

glBindVertexArray(0)

def keyPressed(self, key):
 """keypress handler"""
 if key == 'x':
 self.mode = SliceRender.XSLICE
 # reset slice index
 self.currSliceIndex = int(self.Nx/2)
 self.currSliceMax = self.Nx
 elif key == 'y':
 self.mode = SliceRender.YSLICE
 # reset slice index
 self.currSliceIndex = int(self.Ny/2)
 self.currSliceMax = self.Ny
 elif key == 'z':
 self.mode = SliceRender.ZSLICE
 # reset slice index

```

```

 self.currSliceIndex = int(self.Nz/2)

 self.currSliceMax = self.Nz

 elif key == 'l':

 self.currSliceIndex = (self.currSliceIndex + 1) % self.curr
 SliceMax

 elif key == 'r':

 self.currSliceIndex = (self.currSliceIndex - 1) % self.curr
 SliceMax

 def close(self):

 pass

```

## 11.12 实验

在本节实验中，我们将使用 `volrender.py` 来渲染一个 3D 体积。  
 我们将使用 `RenderWin` 来渲染一个 3D 体积。  
 我们将使用 `9` 和 `10` 来渲染一个 3D 体积。  
 我们将使用 `volrender.py` 来渲染一个 3D 体积。  
 我们将使用 `11.13` 来渲染一个 3D 体积。

在本节实验中，我们将使用 `volrender.py` 来渲染一个 3D 体积。

```

 # load volume data

 ❶ self.volume = volreader.loadVolume(imageDir)

 # create renderer

 ❷ self.renderer = RayCastRender(self.width, self.height, self
 .volume)

```

# ① OpenGL ② RayCastRender

```

 void V(){}
 RenderWindow{}

```

```
def onKeyboard(self, win, key, scancode, action, mods):

print 'keyboard: ', win, key, scancode, action, mods

 # ESC to quit

 if key is glfw.GLFW_KEY_ESCAPE:

 self.renderer.close()

 self.exitNow = True

 else:

❶ if action is glfw.GLFW_PRESS or action is glfw.GLFW_REPEAT:

 if key == glfw.GLFW_KEY_V:

 # toggle render mode

❷ if isinstance(self.renderer, RayCastRender):

self.renderer = SliceRender(self.width, self.height,

 self.volume)

 else:

self.renderer = RayCastRender(self.width, self.height,

 self.volume)
```

```

 # call reshape on renderer

self.renderer.reshape(self.width, self.height)

 else:

 # send keypress to renderer

❸
keyDict = {glfw.GLFW_KEY_X: 'x', glfw.GLFW_KEY_Y: 'y',

glfw.GLFW_KEY_Z: 'z', glfw.GLFW_KEY_LEFT: 'l',

glfw.GLFW_KEY_RIGHT: 'r'}

 try:

 self.renderer.keyPressed(keyDict[key])

 except:

 pass

```

ESC 1  
 2  
 Python isinstance()  
 3

ESC 3  
 keyPressed()

00

glfw.KEY  
 GLFW volrender.py GLFW



```

os.chdir(cwd)

version hints

glfw.glfwWindowHint(glfw.GlfwContextVersionMajor, 3)

glfw.glfwWindowHint(glfw.GlfwContextVersionMinor, 3)

glfw.glfwWindowHint(glfw.GlfwOpenGlForwardCompat, GL_TRUE)

glfw.glfwWindowHint(glfw.GlfwOpenGlProfile,
 glfw.GlfwOpenGlCoreProfile)

make a window

self.width, self.height = 512, 512

self.aspect = self.width/float(self.height)

self.win = glfw.glfwCreateWindow(self.width, self.height, b
"volrender")

make context current

glfw.glfwMakeContextCurrent(self.win)

initialize GL

glViewport(0, 0, self.width, self.height)

glEnable(GL_DEPTH_TEST)

glClearColor(0.0, 0.0, 0.0, 0.0)

```

```
set window callbacks

glfw.glfwSetMouseButtonCallback(self.win, self.onMouseButton)

glfw.glfwSetKeyCallback(self.win, self.onKeyboard)
glfw.glfwSetWindowSizeCallback(self.win, self.onSize)

load volume data

self.volume = volreader.loadVolume(imageDir)

create renderer

self.renderer = RayCastRender(self.width, self.height, self
.volume)

exit flag

self.exitNow = False

def onMouseButton(self, win, button, action, mods):

 # print 'mouse button: ', win, button, action, mods

 pass

def onKeyboard(self, win, key, scancode, action, mods):

 # print 'keyboard: ', win, key, scancode, action, mods
```

```
ESC to quit

if key is glfw.GLFW_KEY_ESCAPE:
 self.renderer.close()
 self.exitNow = True
else:

if action is glfw.GLFW_PRESS or action is glfw.GLFW_REPEAT:
 if key == glfw.GLFW_KEY_V:
 # toggle render mode

if isinstance(self.renderer, RayCastRender):

self.renderer = SliceRender(self.width, self.height,
 self.volume)
 else:

self.renderer = RayCastRender(self.width, self.height,
 self.volume)

 # call reshape on renderer

self.renderer.reshape(self.width, self.height)
 else:

 # send keypress to renderer

keyDict = {glfw.GLFW_KEY_X: 'x', glfw.GLFW_KEY_Y: 'y',
```



```

glfw.GLFW_KEY_Z: 'z', glfw.GLFW_KEY_LEFT: 'l',
 glfw.GLFW_KEY_RIGHT: 'r'}

 try:
 self.renderer.keyPressed(keyDict[key])
 except:
 pass

def onSize(self, win, width, height):
 #print 'onsize: ', win, width, height
 self.width = width
 self.height = height
 self.aspect = width/float(height)
 glViewport(0, 0, self.width, self.height)
 self.renderer.reshape(width, height)

def run(self):
 # start loop

while not glfw.glfwWindowShouldClose(self.win) and not self
.exitNow:

 # render

 self.renderer.draw()

 # swap buffers

```

```

 glfw.glfwSwapBuffers(self.win)

 # wait for events

 glfw.glfwWaitEvents()

 # end

 glfw.glfwTerminate()

main() function
def main():

 print('starting volrender...')

 # create parser

 parser = argparse.ArgumentParser(description="Volume Rendering...")

 # add expected arguments

 parser.add_argument('--
dir', dest='imageDir', required=True)

 # parse args

 args = parser.parse_args()

 # create render window

 rwin = RenderWin(args.imageDir)

 rwin.run()

call main

```

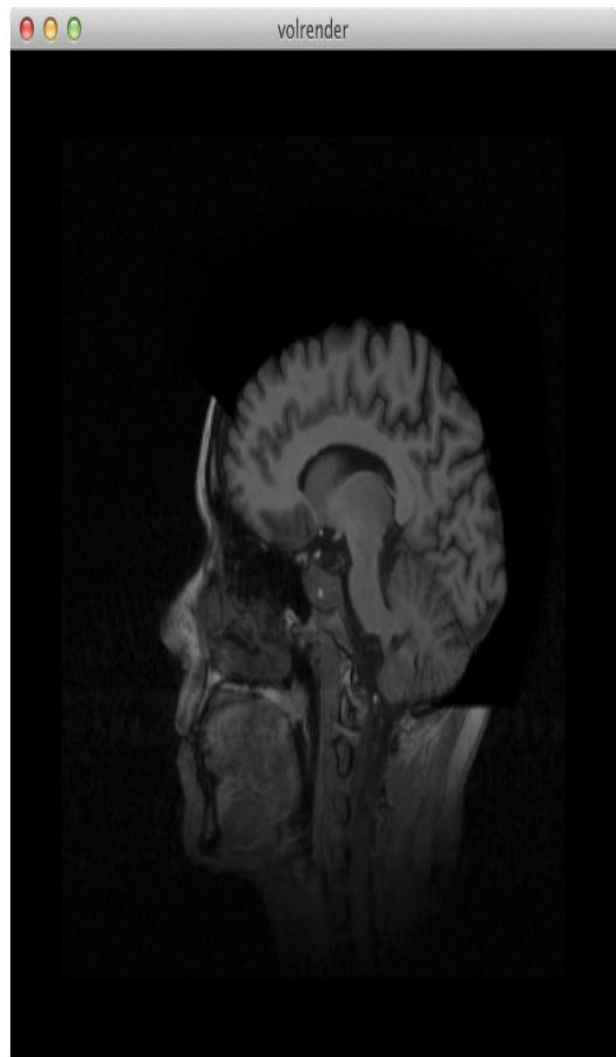
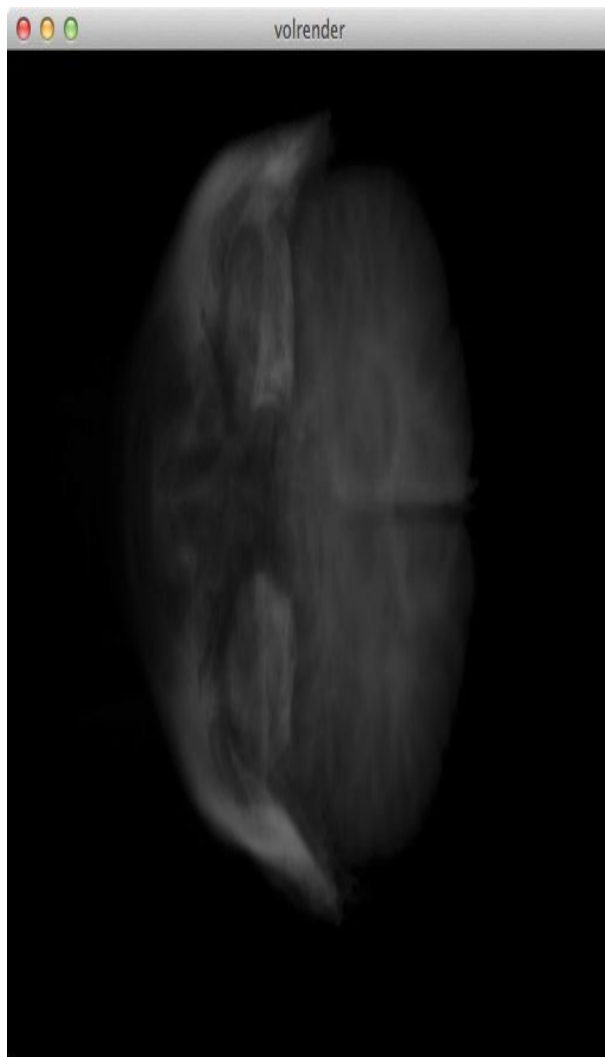
```
if __name__ == '__main__':
 main()
```

## 11.14 □□□□

□□□□□□□□□□□□□□□□ Stanford Volume Data  
Archive□ [\[3\]](#) □□□□□□□□□□□□□□□□

```
$ python volrender.py --dir mrbrain-8bit/
```

□□□□□□ 11-6 □□□□□□



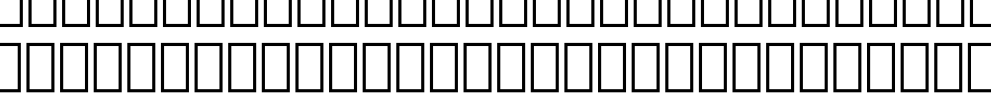

[illegible]

## 11.15

Python OpenGL GLSL

## 11.16

```
1 // ...
WireFrame // ...
x // y // z // ... RayCastRender //
// WireFrame //
```

2   $256 \times 256 \times 99$  

```

3xmaximum intensity
projectionMIP
RayCastRender

```

4. 4D volume rendering: x, y, z, and opacity  
I/O. OpenGL: glutils.lookAt() and  
glutils.projecton() for 3D rendering.

---

[1] J. Kruger and R. Westermann,  
“Acceleration Techniques for GPU-based  
Volume Rendering,” IEEE Visualization,  
2003.

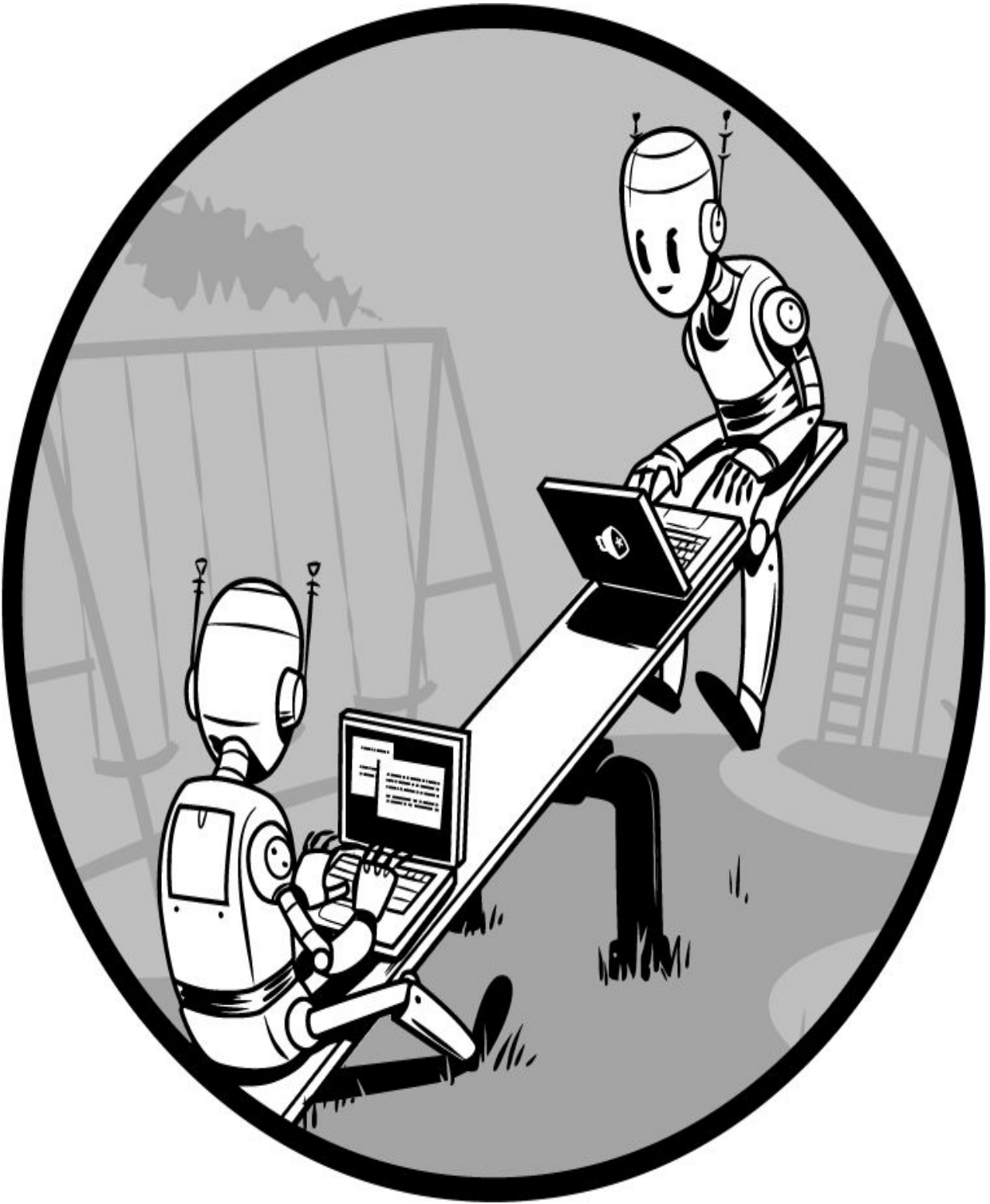
[2]  
<http://graphics.stanford.edu/data/voldata/>

[3]  
<http://graphics.stanford.edu/data/voldata/>

Volume Rendering

“Volume Rendering: A Survey”  
by John Kruger and Rüdiger Westermann

— — —



## 12 Arduino



```
Arduino
Arduino
/

```

Arduino  
Arduino  
C++ Arduino  
Arduino  
pySerial  
Arduino matplotlib  
EKG 12-1

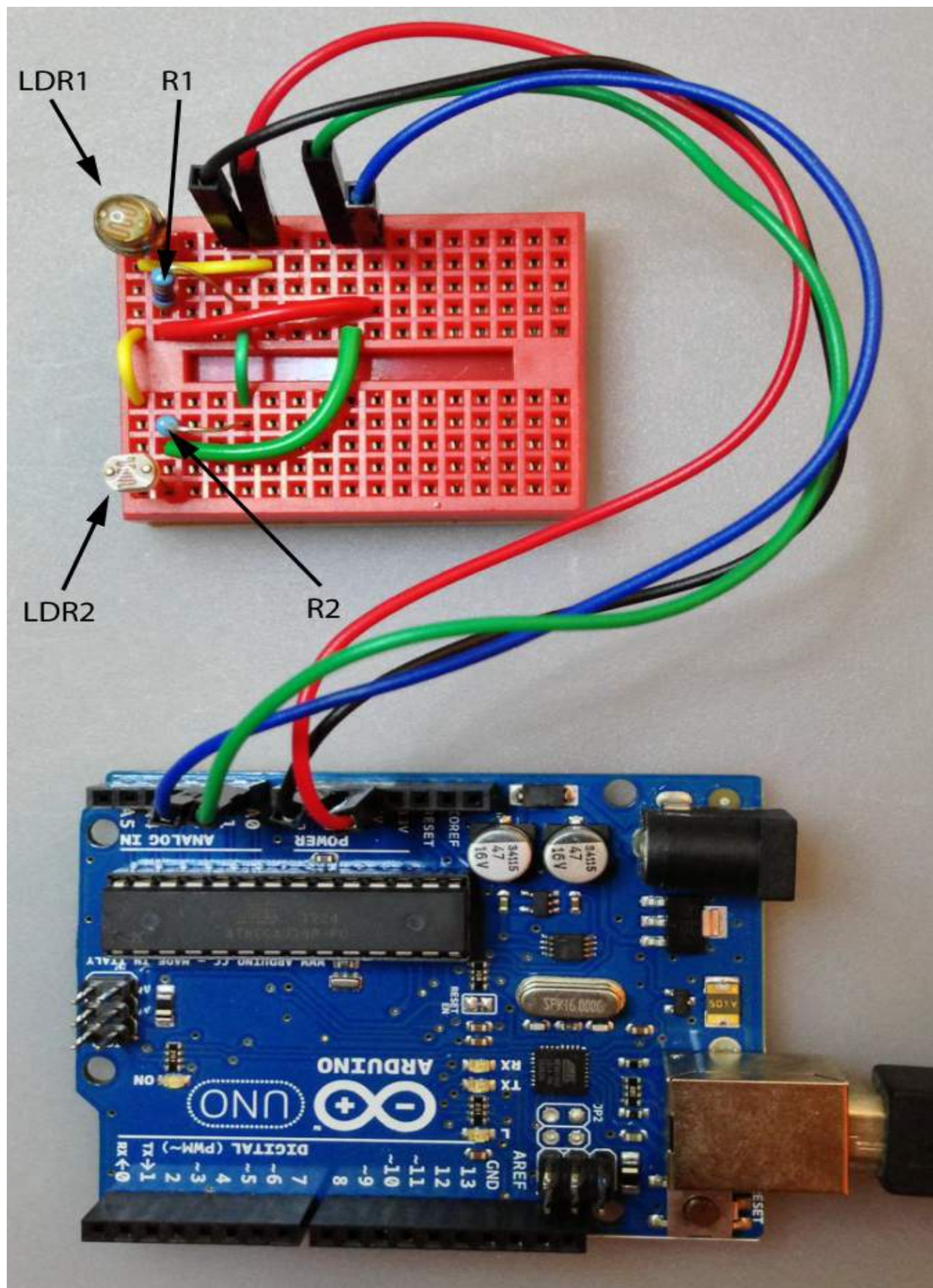




图12-1 使用LDR传感器控制Arduino Uno

## 12.1 Arduino

Arduino是一个基于Atmel AVR微控制器的开源电子原型平台。图12-2展示了Arduino Uno板。Arduino Uno板是一个基于Atmel AVR微控制器的开源电子原型平台。图12-2展示了Arduino Uno板。Arduino Uno板是一个基于Atmel AVR微控制器的开源电子原型平台。图12-2展示了Arduino Uno板。John Boxall的《Arduino Workshop》[No Starch Press, 2013]是一本关于Arduino的书籍。

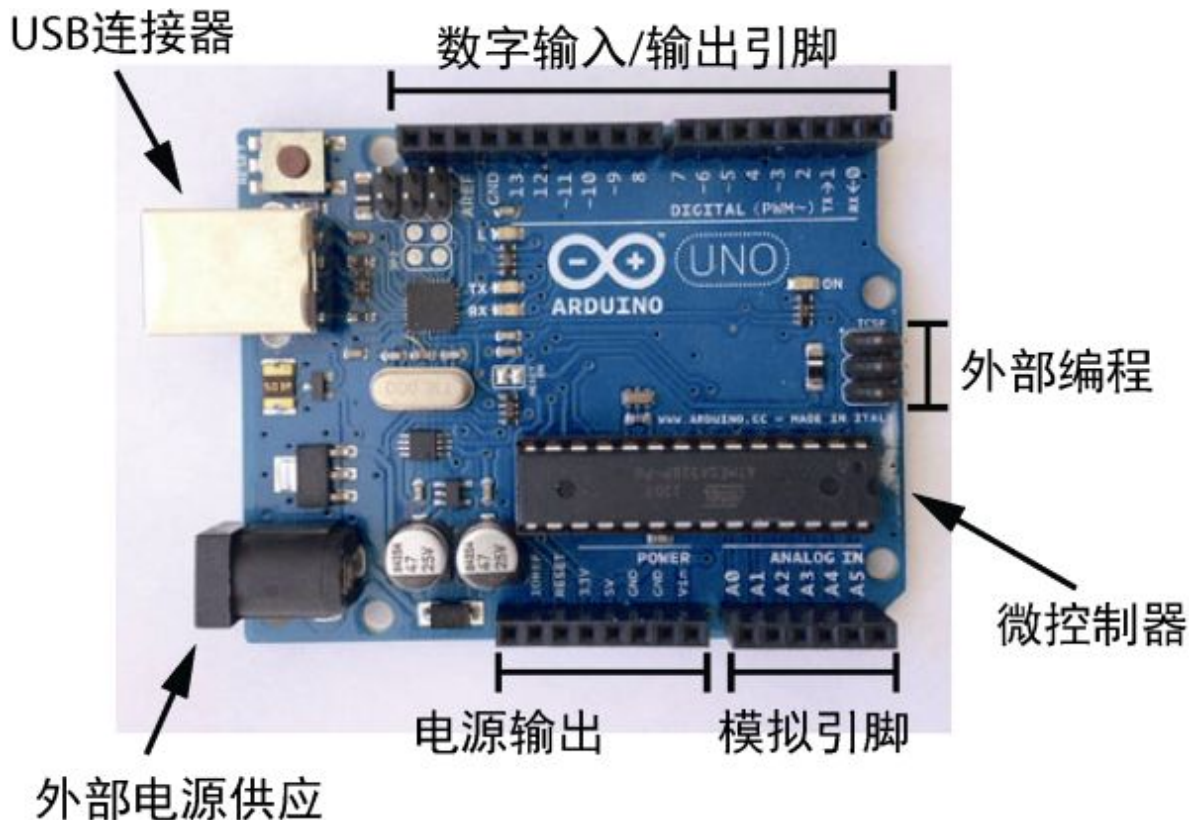


图12-2 Arduino Uno板

Arduino Uno  
Arduino <http://arduino.cc/>  
Uno

Arduino Uno USB  
/

Arduino “bootloader”  
“  
ICSP” ICSP  
Arduino ICSP Arduino  
USB Arduino

Arduino AVR  
Arduino Uno AVR ATmega328  
CPU /  
CPU /  
ADC

## 12.2 Arduino

Arduino  
IDE

## 12.2.1

Arduino 是 C++ 的变体，它结合了 Processing 和 Wiring 的一些特性。Arduino 的源代码存储在 sketches 文件夹中。Arduino 的官方网站是 <http://arduino.cc/>

## 12.2.2 IDE

Arduino 的 IDE 是集成开发环境。Arduino 12-3 的 IDE 包含了一个 serial monitor。Arduino 的 IDE 是免费的，可以在官方网站上找到。IDE 的界面非常友好，适合初学者使用。

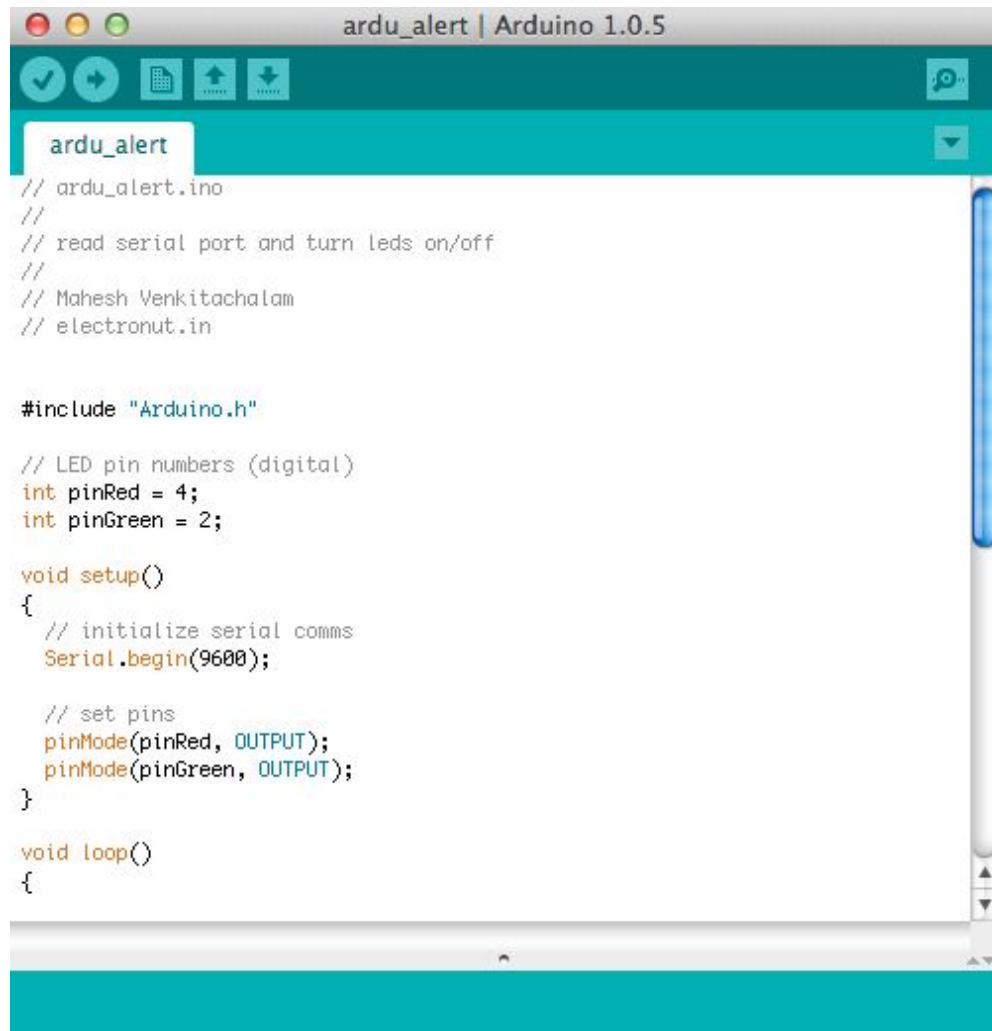


图12-3 Arduino IDE界面截图

### 12.2.3 引脚

Arduino板上有许多引脚，这些引脚用于连接各种电子元件。Arduino板上的引脚分为数字引脚和模拟引脚。数字引脚用于连接数字元件，如LED灯、按钮、继电器等。模拟引脚用于连接模拟元件，如电位器、传感器等。Arduino板上的引脚编号通常从1开始，按顺序排列。在连接元件时，需要注意引脚的极性，特别是对于有极性的元件，如LED灯和电解电容。

### 12.2.4 引脚



ArduinoのGNDは、電源のGNDと共通にする。12-1

## 12.4.1 光センサー

光センサーはLDRと呼ばれる。"光センサー"と呼ばれる。A0に接続する。

$$V_0 = V \frac{R_1}{R_{LDR} + R_1}$$

$R_1$ は定抵抗、 $R_{LDR}$ はLDRの抵抗、 $V$ は電源電圧、 $V_0$ はLDRの抵抗が変化する。A0に接続する。0~5Vの範囲で、A0の値は0~1023の範囲で変化する。

$R_1$ と $R_2$ は定抵抗、LDRは光センサー。LDRの抵抗が変化する。A0に接続する。0V~5Vの範囲で、A0の値は0~1023の範囲で変化する。

4.7kΩの定抵抗、 $R_1$ と $R_2$ は定抵抗、LDRは光センサー。10kΩの定抵抗、1kΩの定抵抗、1.6V~4Vの範囲で、A0の値は0~1023の範囲で変化する。

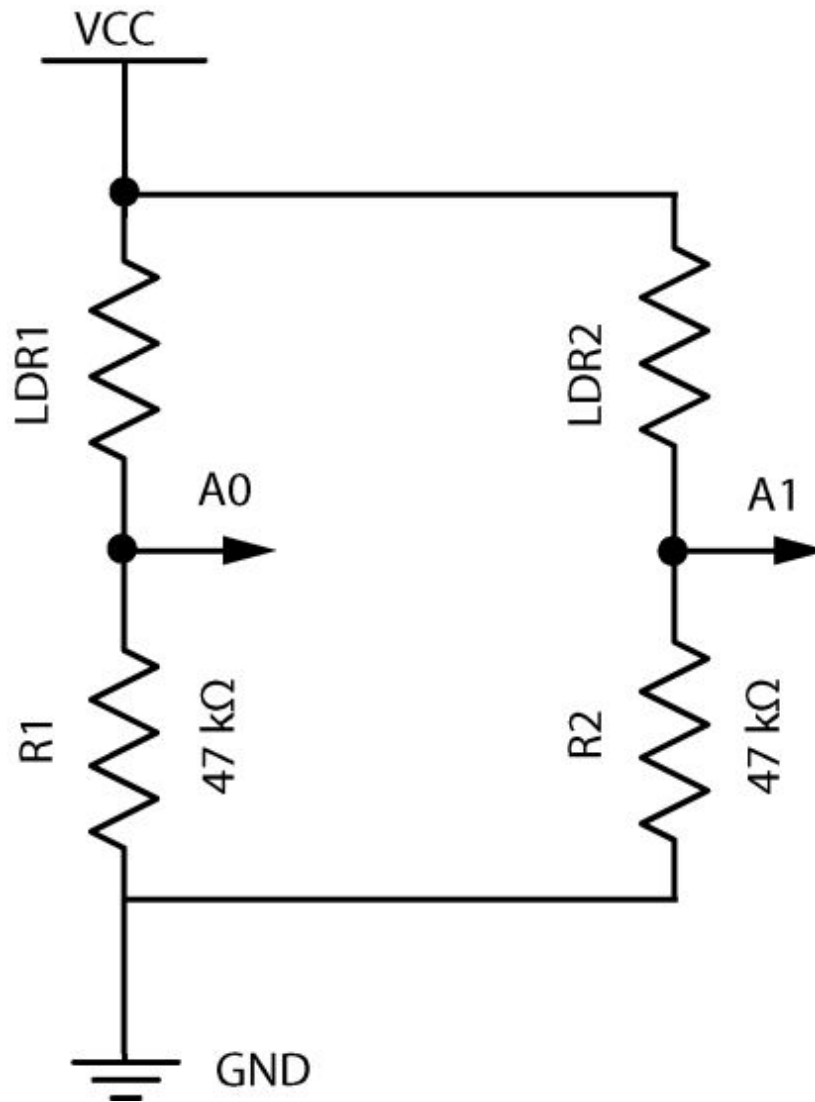


图12-4 光电传感器电路

## 12.4.2 Arduino

Arduino 开发环境安装与配置

```
#include "Arduino.h"
```

```
void setup()
```

```
{
 // initialize serial communications
 ❶ Serial.begin(9600);
}
```

```
void loop()
{
 // read A0
 ❷ int val1 = analogRead(0);
 // read A1
 ❸ int val2 = analogRead(1);
 // print to serial
 ❹ Serial.print(val1);
 Serial.print(" ");
 Serial.print(val2);
 Serial.print("\n");
 // wait
 ❺ delay(50);
}
```

❶ `setup()` `setup()`



[illegible]

```

while (true) {
 [0,1023] random(10) // ➊ Random number 0 to 1023
 Serial.print() // ➋ Print the random number
 delay(50) // ➌ Delay 50ms
 loop() // ➍ Go back to the start of the loop
}

```

```

Arduino IDE
IDE
IDE
Upload
Arduino

```

512 300

513 280

400 200

• • •

```
01 Arduino USB
```

### 12.4.3 思考题

deque 4  
deque N

deque는 리스트와 유사한 자료구조로, 양쪽 끝에서 데이터를 추가하고 삭제할 수 있다.

## 12.5 Python

Python을 사용하여 Arduino를 제어하는 AnalogPlot 클래스를 정의한다.

```
class AnalogPlot:
 # constructor
 def __init__(self, strPort, maxLen):
 # open serial port
 ❶ self.ser = serial.Serial(strPort, 9600)

 ❷ self.a0Vals = deque([0.0]*maxLen)
 ❸ self.a1Vals = deque([0.0]*maxLen)
 ❹ self.maxLen = maxLen
```

❶ AnalogPlot 클래스에서 pySerial 라이브러리의 Serial 클래스를 사용하여 Arduino와 연결한다. Serial 클래스의 init 메서드는 IDE의 Tools -> Serial Port에서 Windows에서는 COM3, Linux에서는 /dev/tty.usbmodem411로 지정된 Arduino의 포트 번호를 입력한다.

② ③ deque maxlen deque ④ maxlen AnalogPlot

deque AnalogPlot

```
add data
```

```
def add(self, data):
```

```
 assert(len(data) == 2)
```

```
 ① self.addToDeq(self.a0Vals, data[0])
```

```
 ② self.addToDeq(self.a1Vals, data[1])
```

```
add to deque; pop oldest value
```

```
def addToDeq(self, buf, val):
```

```
 ③ buf.pop()
```

```
 ④ buf.appendleft(val)
```

Arduino add() ① ② addToDeq() deque ③ ④ pop() deque appendleft() deque deque

matplotlib animation 5 Boids AnalogPlot

`def update()`

`# update plot`

`def update(self, frameNum, a0, a1):`

`try:`

❶ `line = self.ser.readline()`

❷ `data = [float(val) for val in line.split()]`

`# print data`

`if(len(data) == 2):`

❸ `self.add(data)`

❹ `a0.set_data(range(self.maxLen), self.a0Vals)`

❺ `a1.set_data(range(self.maxLen), self.a1Vals)`

`except:`

❻ `pass`

`return a0, a1`

❶ `update()` ❷

Python `split()` 512  
600\n [512,600]

❸ `AnalogPlot.add()`  
❹ `deque` ❺ `matplotlib.set_data()`  
`x[0... maxLen]`  
`range()` `y deque`

```

try:
 pass
except:
 pass
try:
 pass
except:
 pass

```

```

clean up

```

```

def close(self):
 # close serial
 self.ser.flush()
 self.ser.close()

```

```

def main():
 import matplotlib.pyplot as plt

```

```

 # set up animation
 ❶ fig = plt.figure()
 ❷ ax = plt.axes(xlim=(0, maxLen), ylim=(0, 1023))
 ❸ a0, = ax.plot([], [])
 ❹ a1, = ax.plot([], [])
 ❺ anim = animation.FuncAnimation(fig, analogPlot.update,
 fargs=(a0, a1), interval=20)

 # show plot
 ❻ plt.show()

```

① matplotlib figure  
② axes x y x y  
1023

③ ④ a0 a1  
animation  
⑤  
analogPlot update()  
20  
⑥ plt.show()

main() Python argparse

```
create parser

parser = argparse.ArgumentParser(description="LDR serial")

add expected arguments

parser.add_argument('--port', dest='port', required=True)

parser.add_argument('--N', dest='maxLen', required=False)

parse args

args = parser.parse_args()

strPort = args.port
```

```
plot parameters

maxLen = 100

if args.maxLen:

 maxLen = int(args.maxLen)
```

```
--port Arduino
IDE Tools Serial Port maxLen
100
```

## 12.6 Python

Python

<https://github.com/electronut/pp/tree/master/arduino-ldr/ldr.py>

```
import serial, argparse

from collections import deque

import matplotlib.pyplot as plt

import matplotlib.animation as animation
```

```
plot class

class AnalogPlot:

 # constructor

 def __init__(self, strPort, maxLen):

 # open serial port
```

```

self.ser = serial.Serial(strPort, 9600)

self.a0Vals = deque([0.0]*maxLen)
self.a1Vals = deque([0.0]*maxLen)
self.maxLen = maxLen

add data
def add(self, data):
 assert(len(data) == 2)
 self.addToDeq(self.a0Vals, data[0])
 self.addToDeq(self.a1Vals, data[1])

add to deque; pop oldest value
def addToDeq(self, buf, val):
 buf.pop()
 buf.appendleft(val)

update plot
def update(self, frameNum, a0, a1):
 try:
 line = self.ser.readline()
 data = [float(val) for val in line.split()]
 # print data

```



```

 if(len(data) == 2):

 self.add(data)

 a0.set_data(range(self.maxLen), self.a0Vals)

 a1.set_data(range(self.maxLen), self.a1Vals)

 except:

 pass

 return a0, a1

clean up
def close(self):

 # close serial

 self.ser.flush()

 self.ser.close()

main() function
def main():

 # create parser

parser = argparse.ArgumentParser(description="LDR serial")

 # add expected arguments

 parser.add_argument('--
port', dest='port', required=True)

```

```
 parser.add_argument('--
N', dest='maxLen', required=False)

 # parse args

 args = parser.parse_args()

 #strPort = '/dev/tty.usbserial-A7006Yqh'
 strPort = args.port

 print('reading from serial port %s...' % strPort)

 # plot parameters
 maxLen = 100
 if args.maxLen:
 maxLen = int(args.maxLen)

 # create plot object
 analogPlot = AnalogPlot(strPort, maxLen)

 print('plotting data...')

 # set up animation
 fig = plt.figure()
 ax = plt.axes(xlim=(0, maxLen), ylim=(0, 1023))
```

```
a0, = ax.plot([], [])
a1, = ax.plot([], [])
anim = animation.FuncAnimation(fig, analogPlot.update,
 fargs=(a0, a1), interval=20)
```

```
show plot
```

```
plt.show()
```

```
clean up
```

```
analogPlot.close()
```

```
print('exiting.')
```

```
call main
```

```
if __name__ == '__main__':
```

```
 main()
```

## 12.7 □□□□

□□□□□□□□□□LDR□□□□Arduino□□□□□□□□□□□□  
□□□□Python□□□

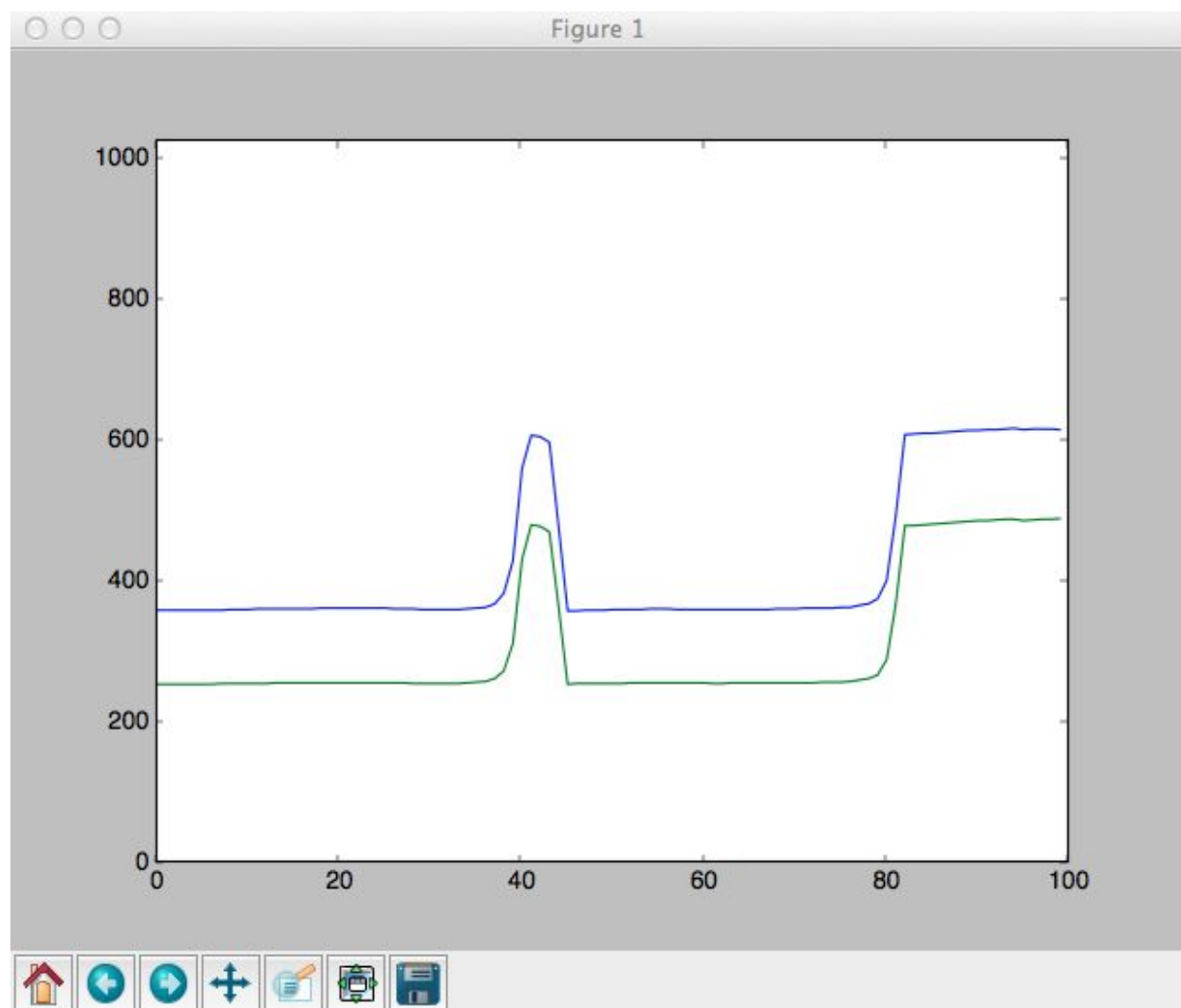
```
$ python3 --port /dev/tty.usbmodem411 ldr.py
```

12-5 光敏电阻的Arduino控制

光敏电阻的Arduino控制

光敏电阻的Arduino控制

光敏电阻的Arduino控制



12-5 光敏电阻的Arduino控制

光敏电阻的Arduino控制

光敏电阻的Arduino控制

## 12.8

Arduino  
Arduino  
Arduino  
LDR  
Arduino  
Python  
matplotlib

## 12.9

Arduino

1

```
2 Arduino
LDR LDR N
delay()
N
```

[illegible]

## 13 物联网



12 物联网 Arduino 物联网  
物联网 Arduino  
Python 物联网  
物联网

物联网  
物联网  
物联网  
物联网  
物联网  
Arduino  
Python Python  
物联网  
物联网

物联网 Arduino Python  
物联网

- 環境構築
- 基礎知識
- numpy
- pyaudio
- Arduino
- Arduino

## 13.1 環境構築

環境構築は、まずPythonのインストールから始める。Python 3.6以上をインストールする。次に、numpyとpyaudioのインストールを行う。numpyは、数値計算のためのライブラリで、pyaudioは、オーディオ処理のためのライブラリである。Arduinoは、マイコンボードの一種で、C++でプログラムを書くことができる。Arduinoのインストールは、Arduino IDEをダウンロードして行う。

次に、13-1の環境構築を行う。Bは、オーディオ処理のためのライブラリである。Bのインストールは、Bのソースコードをダウンロードして行う。

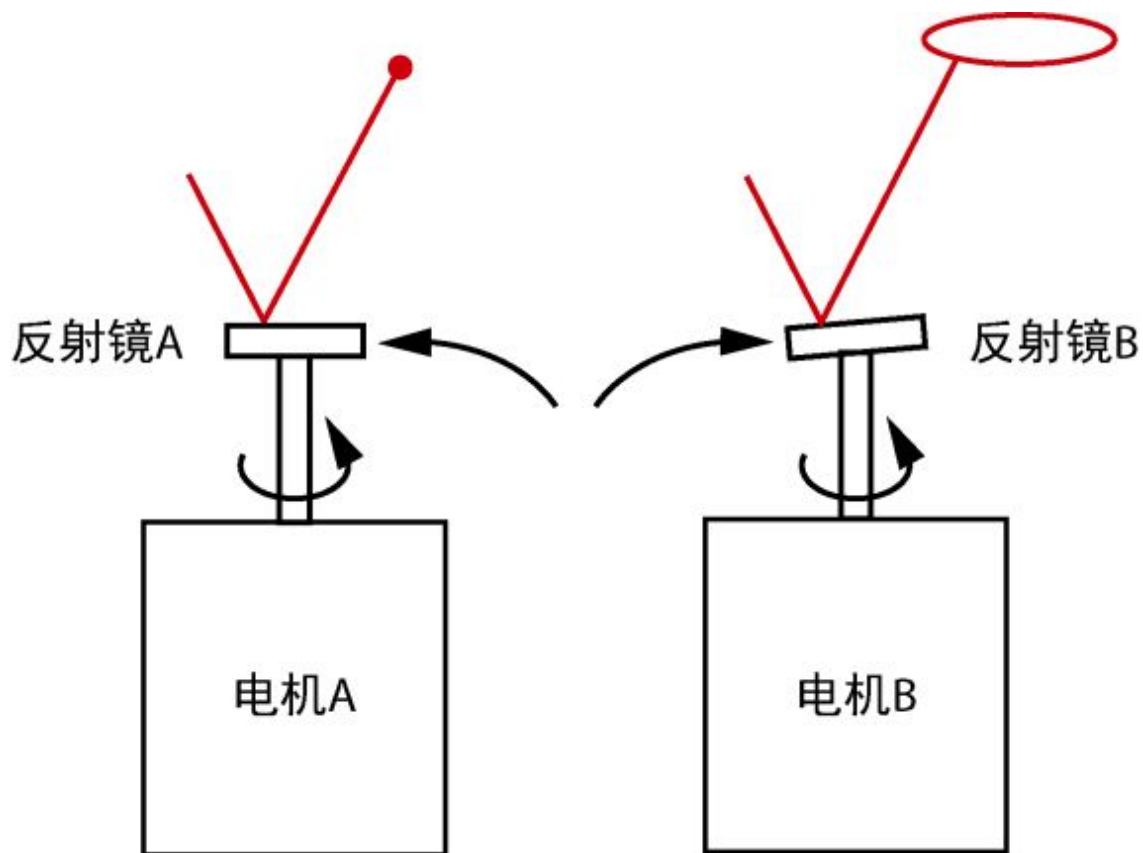


图13-1 基于反射镜A和反射镜B的通信系统

图13-1展示了基于反射镜A和反射镜B的通信系统。该系统由两个电机（电机A和电机B）和两个反射镜（反射镜A和反射镜B）组成。电机A和电机B分别驱动反射镜A和反射镜B。反射镜A和反射镜B通过反射光线进行通信。图13-2展示了该系统的硬件连接图。

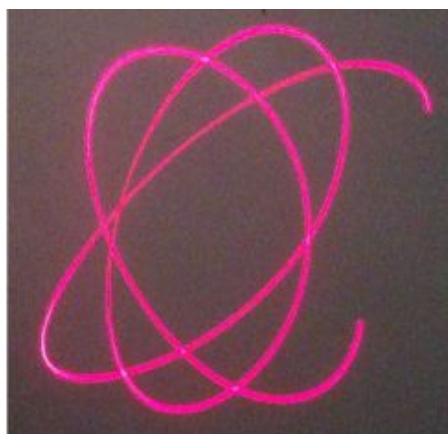
图13-2展示了该系统的硬件连接图。图13-2展示了该系统的硬件连接图。图13-2展示了该系统的硬件连接图。

### 13.1.1 硬件

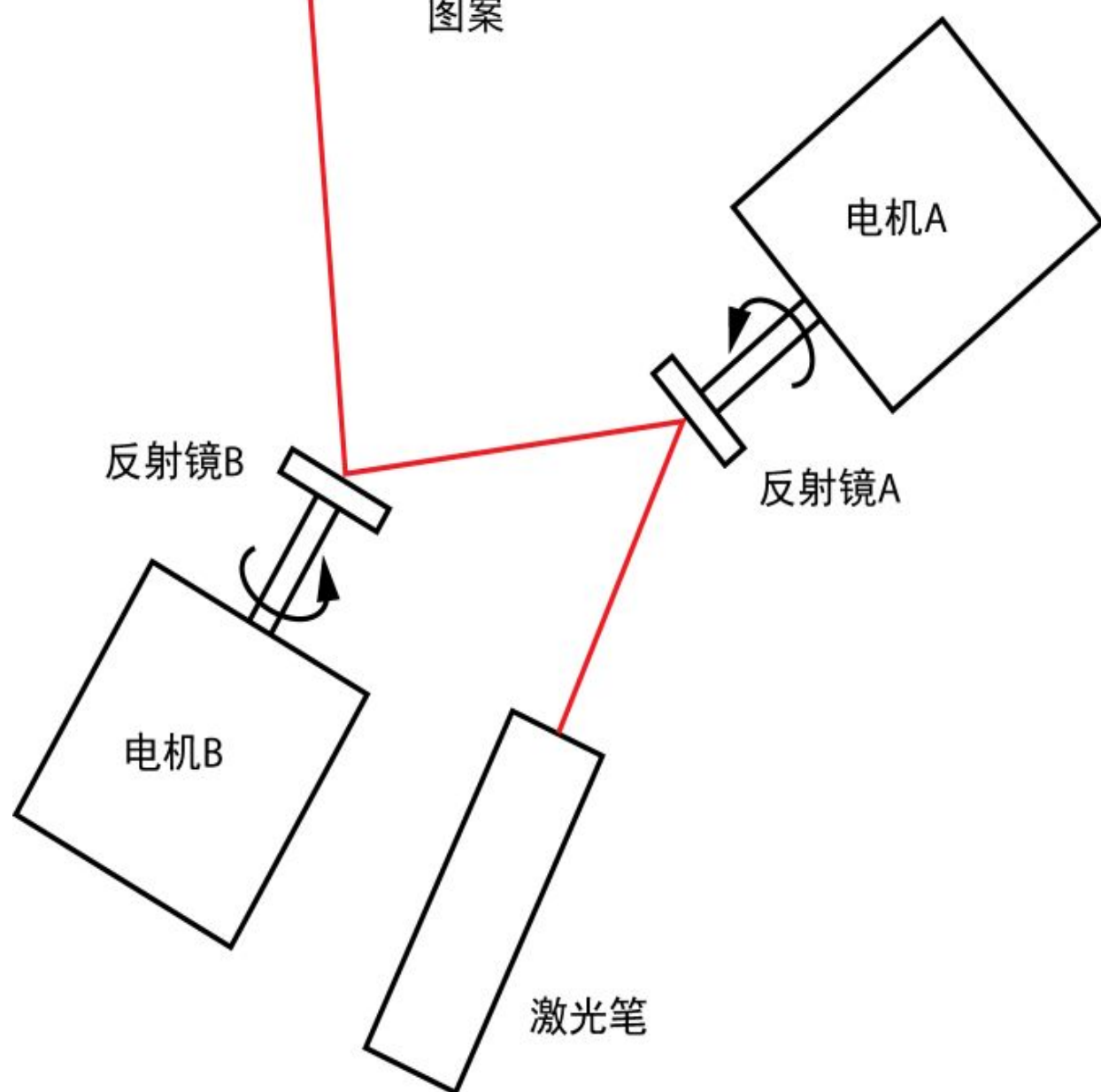
本系统采用Arduino Uno作为主控单元，通过Arduino Uno控制电机A和电机B。图13-3a展示了SparkFun TB6612FNG



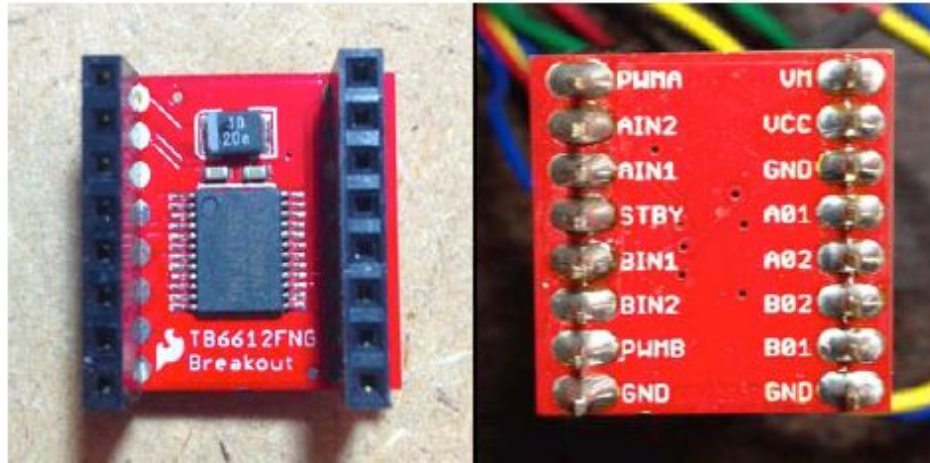
“breakout”Arduino  
Arduino



图案



## 图13-2 引脚分配图



(a)

(b)

## 图13-3 SparkFun 1A 1.2V TB6612FNG

图13-3b 引脚分配图。A、B 引脚为 IN 引脚，01、02 为 PWM 引脚。引脚分配图如下：

图13-3

图13-3b 引脚分配图。A、B 引脚为 IN 引脚，01、02 为 PWM 引脚。引脚分配图如下：

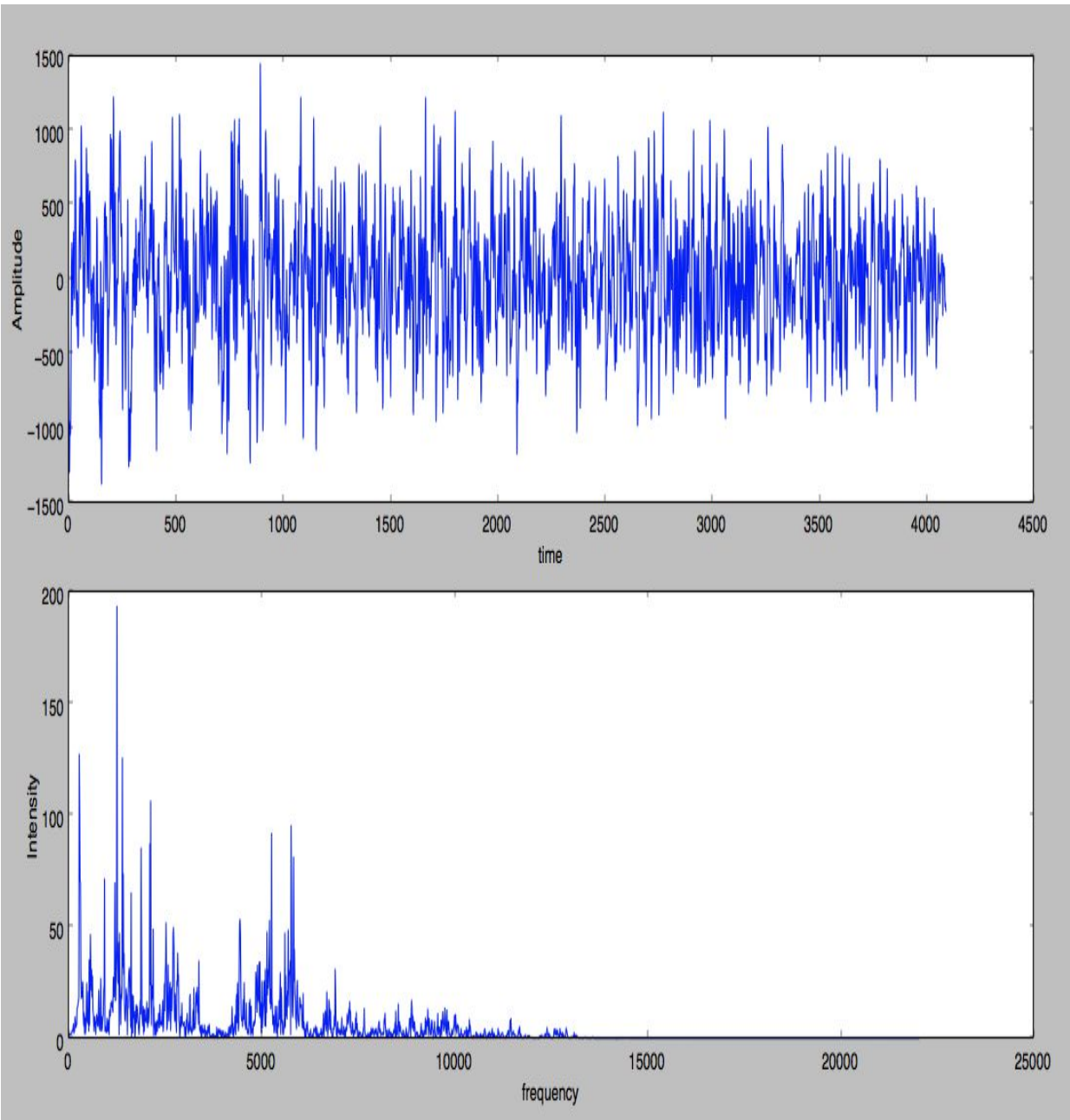
## 13.1.2 引脚分配图

图13-3b 引脚分配图。A、B 引脚为 IN 引脚，01、02 为 PWM 引脚。引脚分配图如下：

4  
 DFT  
 Python  
 FFT  
 DFT  
 FFT

FFT 13-4  
FFT

$$y(t) = 4\sin(2\pi 10t) + 2.5\sin(2\pi 30t)$$

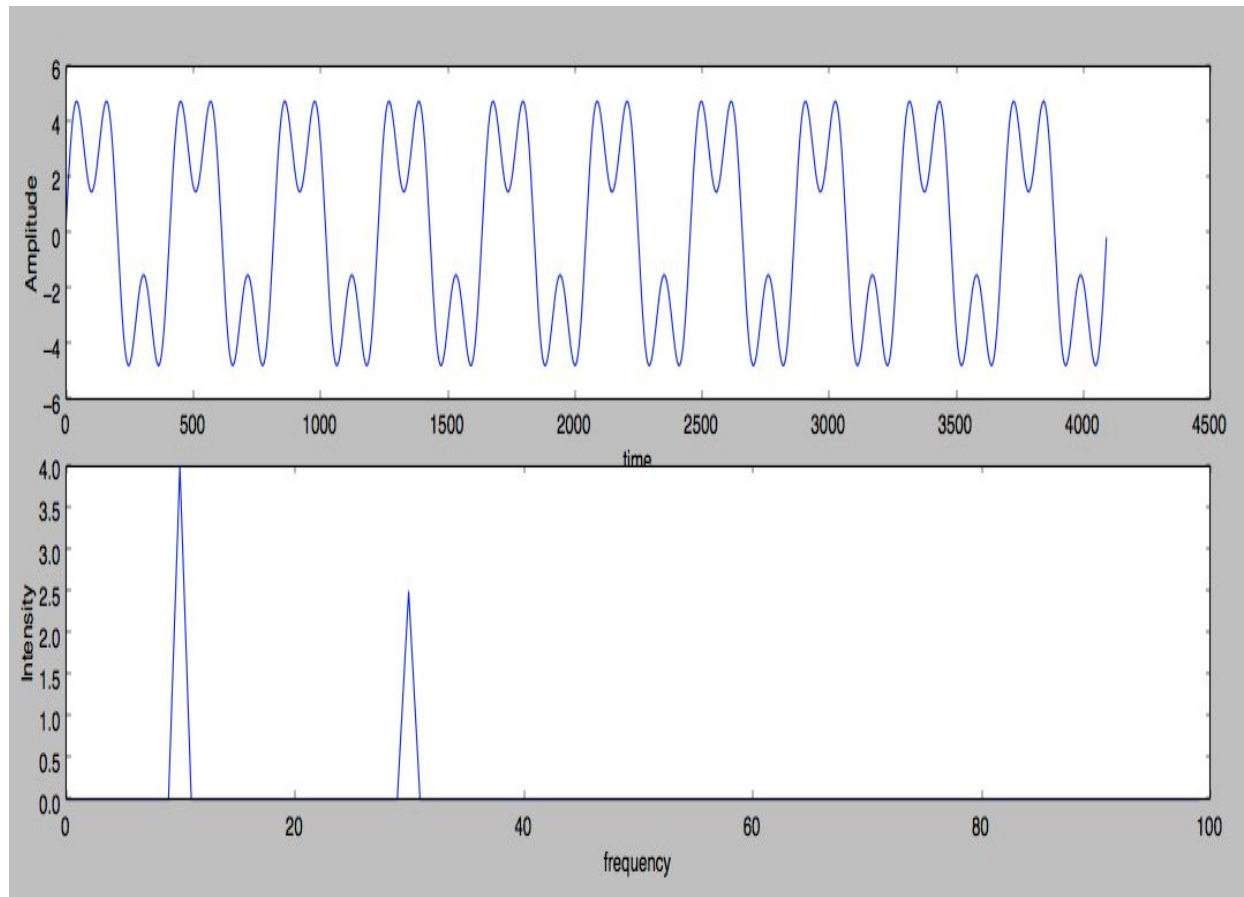


### 13-4 快速傅里叶变换FFT

快速傅里叶变换FFT是离散傅里叶变换的高效实现，其计算复杂度为 $O(N \log N)$ ，其中 $N$ 为采样点数。FFT广泛应用于信号处理、图像处理等领域。

FFT的分辨率与采样率、采样时间密切相关。分辨率越高，频率分辨能力越强。通常，FFT的分辨率在10 Hz到30 Hz之间，具体取决于采样参数。

# 13-5 FFT



## 13-5 FFT

“”FFT“”  
13-5FFT

FFT

“” 44100Hz 2048  
0.046

Hz 3 [0,100]  
[100,1000] [1000,2500]

- 
- 
- 

## 13.2

- 
- 9V
- 2.54
- SparkFun 1A TB6612FNG
- Arduino Uno
- 
- AA
- 20.3 × 15.2
- 
- 
-

## 13.2.1 五五五五五

[illegible]



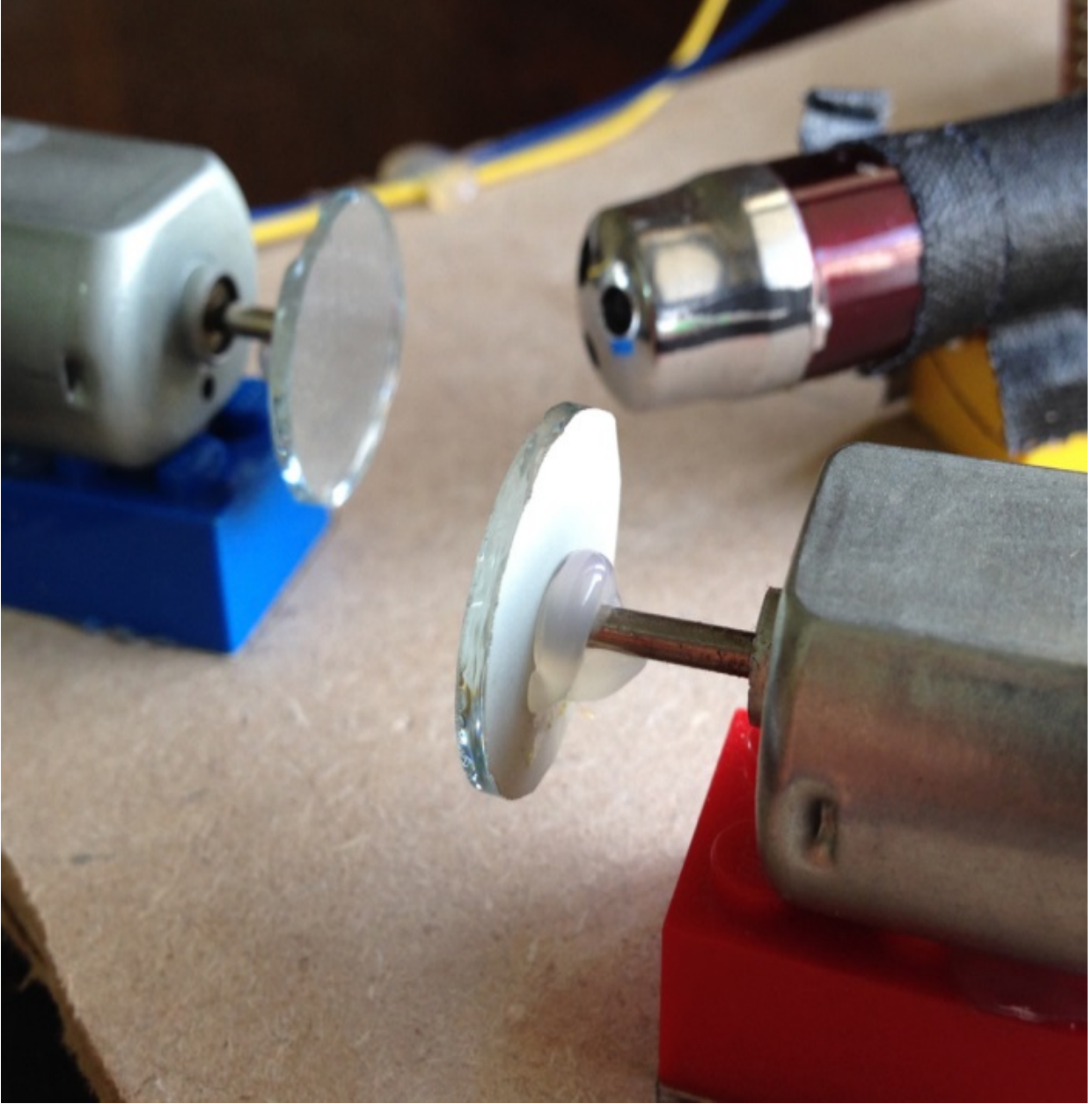


图13-6 实验装置示意图

实验

实验装置如图13-6所示，A为光源，B为光屏，13-7为光路图，A为光源，A为光源，A为光源，B为光屏，B为光屏，B为光屏，A为光源。

A photograph showing two small DC motors mounted on a piece of cardboard. The motor on the left is mounted on a red base and has green and yellow wires. The motor on the right is mounted on a blue base and has blue and yellow wires. A black battery pack with a silver terminal is positioned between the two motors. A black pen is placed vertically next to the battery pack for scale. The wires from the motors are connected to the battery pack.



13.8

[illegible][illegible][illegible]






“H” 是 “MOSFET”

SparkFun Arduino 13-1 A B

### 13-1 SparkFun Arduino

|                                  |             |
|----------------------------------|-------------|
| □                                | □           |
| Arduino Digital Pin 12 TB6612FNG | Pin<br>BIN2 |
| Arduino Digital Pin 11 TB6612FNG | Pin<br>BIN1 |
| Arduino Digital Pin 10 TB6612FNG | Pin<br>STBY |
| Arduino Digital Pin 9 TB6612FNG  | Pin<br>AIN1 |

|                                 |             |
|---------------------------------|-------------|
| Arduino Digital Pin 8 TB6612FNG | Pin<br>AIN2 |
| Arduino Digital Pin 5 TB6612FNG | Pin<br>PWMB |
| Arduino Digital Pin 3 TB6612FNG | Pin<br>PWMA |
| Arduino 5V Pin TB6612FNG        | Pin<br>VCC  |
| Arduino GND TB6612FNG           | Pin<br>GND  |
| Arduino GND Battery Pack        | GND<br>(-)  |
| Battery Pack VCC (+) TB6612FNG  | Pin VM      |
|                                 |             |

|                                                           |                                                                                              |
|-----------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Motor #1 Connector #1 (polarity doesn't matter) TB6612FNG | Pin A01                                                                                      |
| Motor #1 Connector #2 (polarity doesn't matter) TB6612FNG | Pin A02                                                                                      |
| Motor #2 Connector #1 (polarity doesn't matter) TB6612FNG | Pin B01                                                                                      |
| Motor #2 Connector #2 (polarity doesn't matter) TB6612FNG | Pin B02                                                                                      |
| Arduino USB connector                                     | <br>USB |

13-9

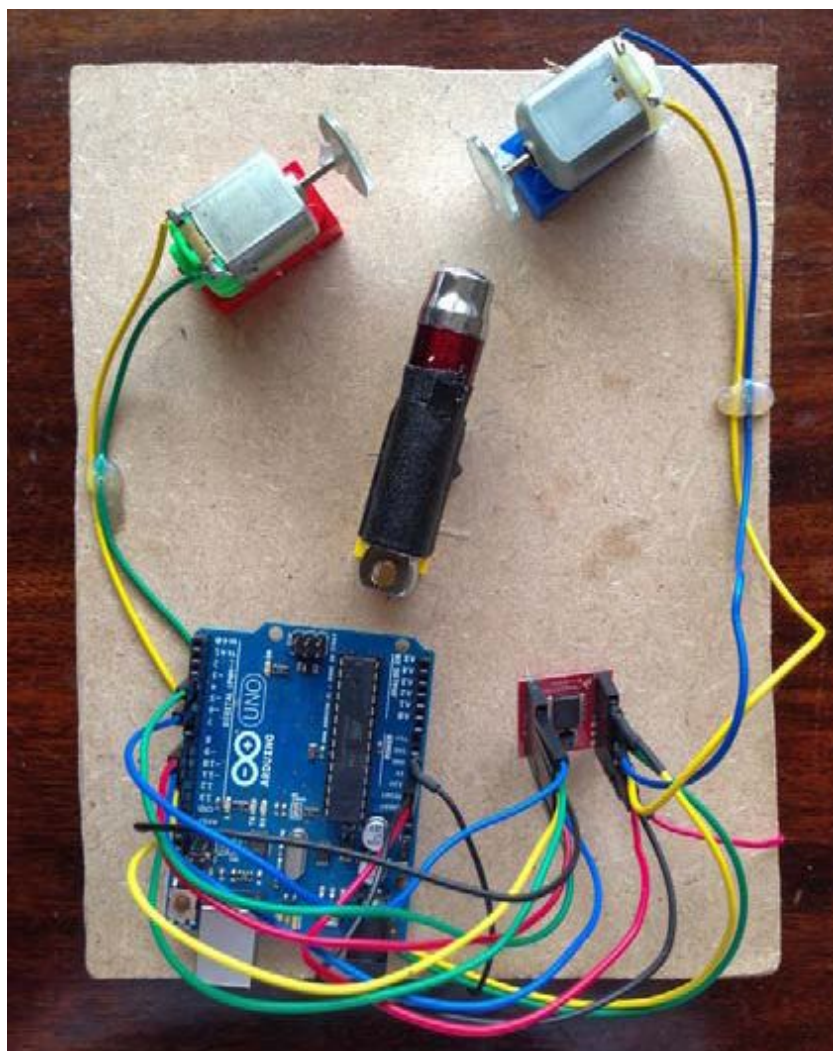


图13-9 电机驱动电路

Arduino 开发板

## 13.3 Arduino 开发板

Arduino 开发板是一种基于 ATmega328P 微控制器的开源电子平台。它由一块电路板组成，上面安装了一个 ATmega328P 微控制器、一个 USB 接口、一个 DC 电源接口、一个复位按钮以及许多其他电子元件。Arduino 开发板可以通过 USB 接口连接到计算机，通过串口（UART）接口与外部设备通信。它还可以通过 I2C、SPI 等接口与各种传感器和执行器连接。Arduino 开发板具有易于使用、成本低廉、社区支持广泛等优点，广泛应用于教育、科研、工业控制等领域。

### 13.3.1 Arduino 开发板



## 13-1 Arduino

```
// motor A connected to A01 and A02
// motor B connected to B01 and B02
```

```
❶ int STBY = 10; //standby
```

```
// Motor A
int PWMA = 3; //speed control
int AIN1 = 9; //direction
int AIN2 = 8; //direction
```

```
// Motor B
int PWMB = 5; //speed control
int BIN1 = 11; //direction
❷ int BIN2 = 12; //direction
```

```
void setup(){
```

```
❸ pinMode(STBY, OUTPUT);
```

```
pinMode(PWMA, OUTPUT);
```

```
pinMode(AIN1, OUTPUT);
```

```
pinMode(AIN2, OUTPUT);
```

```
pinMode(PWMB, OUTPUT);
```

```
pinMode(BIN1, OUTPUT);
```

```
pinMode(BIN2, OUTPUT);
```

```
// initialize serial communication
```

```
④ Serial.begin(9600);
```

```
}
```

① ② Arduino 3 PWM A A Arduino 3 PWM “ ” “ ” PWM LED

③ setup() 7 ④ Arduino

## 13.3.2

```

// main loop that reads the motor data sent by laser.py
void loop()
{
 // data sent is of the form 'H' (header), speed1, dir1,
 speed2, dir2

 ❶ if (Serial.available() >= 5) {
 ❷ if(Serial.read() == 'H') {
 // read the next 4 bytes

 ❸ byte s1 = Serial.read();
 byte d1 = Serial.read();
 byte s2 = Serial.read();
 byte d2 = Serial.read();

 // stop the motor if both speeds are 0

 ❹ if(s1 == 0 && s2 == 0) {
 stop();
 }
 else {
 // set the motors' speed and direction

 ❺ move(0, s1, d1);
 move(1, s2, d2);
 }

 // slight pause for 20 ms

```

```

⑥ delay(20);

 }

 else {

 // if there is invalid data, stop the motors

⑦ stop();

 }

}

else {

 // if there is no data, pause for 250 ms

⑧ delay(250);

 }

}

```

5. H4 s1 d1 s2  
 d2 ①  
 5 250 ⑧  
 ②

② H ④ ⑦  
 ③

③ ④ ⑤ move() ⑥  
 ⑦

void move()

```
// set motor speed and direction
// motor: A -> 1, B -> 0
// direction: 1/0
void move(int motor, int speed, int direction)
{
 // disable standby
 ❶ digitalWrite(STBY, HIGH);

 ❷ boolean inPin1 = LOW;
 boolean inPin2 = HIGH;

 ❸ if(direction == 1){
 inPin1 = HIGH;
 inPin2 = LOW;
 }

 if(motor == 1){
 ❹ digitalWrite(AIN1, inPin1);
 digitalWrite(AIN2, inPin2);
 analogWrite(PWMA, speed);
 }
 else{
```



```
}
```

## 13.4 Python

Python은 FFT를 사용하여 Arduino 13.5를  
처리하는 데 사용됩니다.

### 13.4.1

pyaudio 라이브러리를 사용하여  
pyaudio를

```
p = pyaudio.PyAudio()
```

pyaudio 라이브러리를 사용하여  
getInputDevice()를 사용하여

```
get pyaudio input device
```

```
def getInputDevice(p):
```

```
 ❶ index = None
```

```
 ❷ nDevices = p.get_device_count()
```

```
print('Found %d devices. Select input device:' % nDevices)
```

```
 # print all devices found
```

```
 for i in range(nDevices):
```

```
 ❸ deviceInfo = p.get_device_info_by_index(i)
```

```
 ❹ devName = deviceInfo['name']
```

```

❶ print("%d: %s" % (i, devName))

 # get user selection

 try:

 # convert to integer

❷ index = int(input())

 except:

 pass

 # print the name of the chosen device

 if index is not None:

 devName = p.get_device_info_by_index(index)["name"]

 print("Input device chosen: %s" % devName)

❸ return index

```

```

❶ if index is None:
 return index
❷ else:
 if index is None:
 return None
 else:
 get_device_count()
 # ... (omitted code) ...
 return index

```

```

❸ get_device_info_by_index(index)
 # ... (omitted code) ...
 return info
❹ if index is None:
 return None
❺ else:
 get_device_count()
 # ... (omitted code) ...
 return index
❻ input()
 # ... (omitted code) ...
 return index

```



## 13.4.2 音频输入

在while循环中，我们使用pyaudio库来读取音频数据。以下代码展示了如何设置FFT参数并打开音频流。

```
set FFT sample length
❶ fftLen = 2**11

set sample rate
❷ sampleRate = 44100

print('opening stream...')
❸ stream = p.open(format = pyaudio.paInt16,
 channels = 1,
 rate = sampleRate,
 input = True,
 frames_per_buffer = fftLen,
 input_device_index = inputIndex)
```

❶ 设置FFT长度为2048，即2<sup>11</sup>。❷ 设置采样率为44100 Hz，即44.1 kHz。❸ 使用pyaudio库打开音频流，格式为paInt16，通道数为1，采样率为sampleRate，帧数为frames\_per\_buffer，输入设备索引为inputIndex。

在pyaudio库中，我们使用以下参数来配置音频流：

- `pyaudio.paInt16`：16位整数格式
- `channels`：通道数，这里设置为1

- rate 44100 Hz
- input True
- frames\_per\_buffer FFT
- input\_device\_index  
getInputDevice()

### 13.4.3 FFT

```
read a chunk of data

❶ data = stream.read(fftLen)

convert the data to a numpy array

❷ dataArray = numpy.frombuffer(data, dtype=numpy.int16)
```

❶ fftLen ❷ 16 numpy

FFT

```
get FFT of data

❶ fftVals = numpy.fft.rfft(dataArray)*2.0/fftLen

get absolute values of complex numbers

❷ fftVals = numpy.abs(fftVals)
```

❶ numpy fft rfft() numpy  
FFT “” FFT  
“” 2.0/fftLen FFT

np.fft.rfft() numpy  
abs() ❷

## 13.4.4 FFT

FFT

```
average 3 frequency bands: 0-100 Hz, 100-
1000 Hz, and 1000-2500 Hz
```

```
levels = [numpy.sum(fftVals[0:100])/100,
 numpy.sum(fftVals[100:1000])/900,
 numpy.sum(fftVals[1000:2500])/1500]
```

3 0 100 Hz 100  
1000 Hz 1000 2500 Hz  
0-100 Hz 100-1000 Hz  
numpy.sum()  
FFT

## 13.4.5

```
'H' (header), speed1, dir1, speed2, dir2
```

```
❶ vals = [ord('H'), 100, 1, 100, 1]
```

```
speed1
```

```
❷ vals[1] = int(5*levels[0]) % 255
```

```
speed2
```



## 13.4.6 自動テスト

自動テストを実行するための関数 `autoTest()` を定義する。

```
automatic test for sending motor speeds

def autoTest(ser):
 print('starting automatic test...')
 try:
 while True:
 # for each direction combination
 ❶ for dr in [(0, 0), (1, 0), (0, 1), (1, 1)]:
 # for a range of speeds
 ❷ for j in range(25, 180, 10):
 ❸ for i in range(25, 180, 10):
 ❹ vals = [ord('H'), i, dr[0], j, dr[1]]
 ❺ print(vals[1:])
 ❻ data = struct.pack('BBBBB', *vals)
 ❼ ser.write(data)
 sleep(0.1)
 except KeyboardInterrupt:
 print('exiting...')
 # shut off motors
 ❽ vals = [ord('H'), 0, 1, 0, 1]
```

```
data = struct.pack('BBBBB', *vals)

ser.write(data)

ser.close()
```

1. 5 bytes of data (4 bytes of data + 1 byte of padding)
 2. 1 byte of padding
 3. 1 byte of padding
 4. 1 byte of padding

5. 1 byte of padding

6. 1 byte of padding
 7. 1 byte of padding
 8. 1 byte of padding
 9. 1 byte of padding
 10. 1 byte of padding
 11. 1 byte of padding
 12. 1 byte of padding
 13. 1 byte of padding
 14. 1 byte of padding
 15. 1 byte of padding
 16. 1 byte of padding
 17. 1 byte of padding
 18. 1 byte of padding
 19. 1 byte of padding
 20. 1 byte of padding
 21. 1 byte of padding
 22. 1 byte of padding
 23. 1 byte of padding
 24. 1 byte of padding
 25. 1 byte of padding
 26. 1 byte of padding
 27. 1 byte of padding
 28. 1 byte of padding
 29. 1 byte of padding
 30. 1 byte of padding
 31. 1 byte of padding
 32. 1 byte of padding
 33. 1 byte of padding
 34. 1 byte of padding
 35. 1 byte of padding
 36. 1 byte of padding
 37. 1 byte of padding
 38. 1 byte of padding
 39. 1 byte of padding
 40. 1 byte of padding
 41. 1 byte of padding
 42. 1 byte of padding
 43. 1 byte of padding
 44. 1 byte of padding
 45. 1 byte of padding
 46. 1 byte of padding
 47. 1 byte of padding
 48. 1 byte of padding
 49. 1 byte of padding
 50. 1 byte of padding
 51. 1 byte of padding
 52. 1 byte of padding
 53. 1 byte of padding
 54. 1 byte of padding
 55. 1 byte of padding
 56. 1 byte of padding
 57. 1 byte of padding
 58. 1 byte of padding
 59. 1 byte of padding
 60. 1 byte of padding
 61. 1 byte of padding
 62. 1 byte of padding
 63. 1 byte of padding
 64. 1 byte of padding
 65. 1 byte of padding
 66. 1 byte of padding
 67. 1 byte of padding
 68. 1 byte of padding
 69. 1 byte of padding
 70. 1 byte of padding
 71. 1 byte of padding
 72. 1 byte of padding
 73. 1 byte of padding
 74. 1 byte of padding
 75. 1 byte of padding
 76. 1 byte of padding
 77. 1 byte of padding
 78. 1 byte of padding
 79. 1 byte of padding
 80. 1 byte of padding
 81. 1 byte of padding
 82. 1 byte of padding
 83. 1 byte of padding
 84. 1 byte of padding
 85. 1 byte of padding
 86. 1 byte of padding
 87. 1 byte of padding
 88. 1 byte of padding
 89. 1 byte of padding
 90. 1 byte of padding
 91. 1 byte of padding
 92. 1 byte of padding
 93. 1 byte of padding
 94. 1 byte of padding
 95. 1 byte of padding
 96. 1 byte of padding
 97. 1 byte of padding
 98. 1 byte of padding
 99. 1 byte of padding
 100. 1 byte of padding

4. 5 bytes of data (4 bytes of data + 1 byte of padding)
 5. 1 byte of padding
 Python vals [1]

6. 1 byte of padding
 7. 1 byte of padding
 Ctrl-C
 8. 1 byte of padding

## 13.4.7 argparse

argparse

```
main method

def main():

 # parse arguments
```

```
parser = argparse.ArgumentParser(description='Analyzes audio input and
```

```
sends motor control information via serial port')
```

```
 # add arguments
```

```
 parser.add_argument('--port', dest='serial_port_name', required=True)
```

```
 parser.add_argument('--mtest', action='store_true', default=False)
```

```
 parser.add_argument('--atest', action='store_true', default=False)
```

```
 args = parser.parse_args()
```

```


```

```
def main():
```

```
 # open serial port
```

```
 strPort = args.serial_port_name
```

```
 print('opening ', strPort)
```

```
 ❶ ser = serial.Serial(strPort, 9600)
```

```
 if args.mtest:
```

```
 manualTest(ser)
```

```
 elif args.atest:
```

```
 autoTest(ser)
```

```
 else:
```

```
 ❷ fftLive(ser)
```

❶ `pySerial` 라이브러리 설치  
9600 baudrate로 설정하고 `--atest` 또는 `--mtest` 옵션으로  
❷ `FFT` 라이브러리 `fftLive()` 호출

## 13.4.8 수동 테스트

수동 테스트를 위한 코드는 다음과 같다.

```
manual test of motor direction and speeds

def manualTest(ser):

 print('starting manual test...')

 try:

 while True:

 print('enter motor control info such as < 100 1 120 0 >')

 ❶ strIn = raw_input()

 ❷ vals = [int(val) for val in strIn.split()[:4]]

 ❸ vals.insert(0, ord('H'))

 ❹ data = struct.pack('BBBBB', *vals)

 ❺ ser.write(data)

 except:

 print('exiting...')

 # shut off the motors

 ❻ vals = [ord('H'), 0, 1, 0, 1]
```



```
data = struct.pack('BBBBB', *vals)

ser.write(data)

ser.close()
```

① raw\_input() 100 1 120 0 A B ② ③ "H" ④ ⑤ Ctrl-C ⑥

## 13.5 Python

Python <https://github.com/electronut/pp/tree/master/arduino-laser/laser.py>

```
import sys, serial, struct

import pyaudio

import numpy

import math

from time import sleep

import argparse

manual test of motor direction speeds

def manualTest(ser):
```

```

print('staring manual test...')

try:
 while True:

print('enter motor control info: eg. < 100 1 120 0 >')

 strIn = raw_input()

 vals = [int(val) for val in strIn.split()[:4]]

 vals.insert(0, ord('H'))

 data = struct.pack('BBBBB', *vals)

 ser.write(data)

except:

 print('exiting...')

 # shut off motors

 vals = [ord('H'), 0, 1, 0, 1]

 data = struct.pack('BBBBB', *vals)

 ser.write(data)

 ser.close()

automatic test for sending motor speeds
def autoTest(ser):

 print('staring automatic test...')

 try:

 while True:

 # for each direction combination

```

```

 for dr in [(0, 0), (1, 0), (0, 1), (1, 1)]:
 # for a range of speeds
 for j in range(25, 180, 10):
 for i in range(25, 180, 10):
 vals = [ord('H'), i, dr[0], j, dr[1]]
 print(vals[1:])
 data = struct.pack('BBBBB', *vals)
 ser.write(data)
 sleep(0.1)
except KeyboardInterrupt:
 print('exiting...')
 # shut off motors
 vals = [ord('H'), 0, 1, 0, 1]
 data = struct.pack('BBBBB', *vals)
 ser.write(data)
 ser.close()

get pyaudio input device
def getInputDevice(p):
 index = None
 nDevices = p.get_device_count()

 print('Found %d devices. Select input device:' % nDevices)

```

```

print all devices found
for i in range(nDevices):
 deviceInfo = p.get_device_info_by_index(i)
 devName = deviceInfo['name']
 print("%d: %s" % (i, devName))

get user selection
try:
 # convert to integer
 index = int(input())
except:
 pass

print the name of the chosen device
if index is not None:
 devName = p.get_device_info_by_index(index)["name"]
 print("Input device chosen: %s" % devName)

return index

FFT of live audio
def fftLive(ser):
 # initialize pyaudio
 p = pyaudio.PyAudio()

```

```
get pyAudio input device index
inputIndex = getInputDevice(p)

set FFT sample length
fftLen = 2**11

set sample rate
sampleRate = 44100

print('opening stream...')
stream = p.open(format = pyaudio.paInt16,
 channels = 1,
 rate = sampleRate,
 input = True,
 frames_per_buffer = fftLen,
 input_device_index = inputIndex)

try:
 while True:
 # read a chunk of data
 data = stream.read(fftLen)

 # convert to numpy array

dataArray = numpy.frombuffer(data, dtype=numpy.int16)
```

```

get FFT of data

fftVals = numpy.fft.rfft(dataArray)*2.0/fftLen

get absolute values of complex numbers

fftVals = numpy.abs(fftVals)

average 3 frequency bands: 0-100 Hz, 100-
1000 Hz and 1000-2500 Hz

levels = [numpy.sum(fftVals[0:100])/100,
 numpy.sum(fftVals[100:1000])/900,
 numpy.sum(fftVals[1000:2500])/1500]

the data sent is of the form:

'H' (header), speed1, dir1, speed2, dir2
vals = [ord('H'), 100, 1, 100, 1]

speed1

vals[1] = int(5*levels[0]) % 255

speed2

vals[3] = int(100 + levels[1]) % 255

dir

d1 = 0

if levels[2] > 0.1:
 d1 = 1

```

```
 vals[2] = d1
 vals[4] = 0

 # pack data
 data = struct.pack('BBBBB', *vals)

 # write data to serial port
 ser.write(data)

 # a slight pause
 sleep(0.001)
except KeyboardInterrupt:
 print('stopping...')
finally:
 print('cleaning up')
 stream.close()
 p.terminate()

 # shut off motors
 vals = [ord('H'), 0, 1, 0, 1]
 data = struct.pack('BBBBB', *vals)
 ser.write(data)

 # close serial
 ser.flush()
 ser.close()
```

```

main method

def main():

 # parse arguments

 parser = argparse.ArgumentParser(description='Analyzes audio input and
sends motor control information via serial port')

 # add arguments

 parser.add_argument('--
port', dest='serial_port_name', required=True)

 parser.add_argument('--
mtest', action='store_true', default=False)

 parser.add_argument('--
atest', action='store_true', default=False)

 args = parser.parse_args()

 # open serial port

 strPort = args.serial_port_name

 print('opening ', strPort)

 ser = serial.Serial(strPort, 9600)

 if args.mtest:

 manualTest(ser)

 elif args.atest:

 autoTest(ser)

 else:

```



```
call main function

if __name__ == '__main__':
 main()
```

Arduino

Arduino

Arduino

■ ■ ■

[illegible]

```
$ python3 laser.py --port /dev/tty.usbmodem411
```

```
('opening ', '/dev/tty.usbmodem1411')
```

Found 4 devices. Select input device:

0: Built-in Microph

## 1: Built-in Output

## 2: BoomDevice

### 3: AirParrot

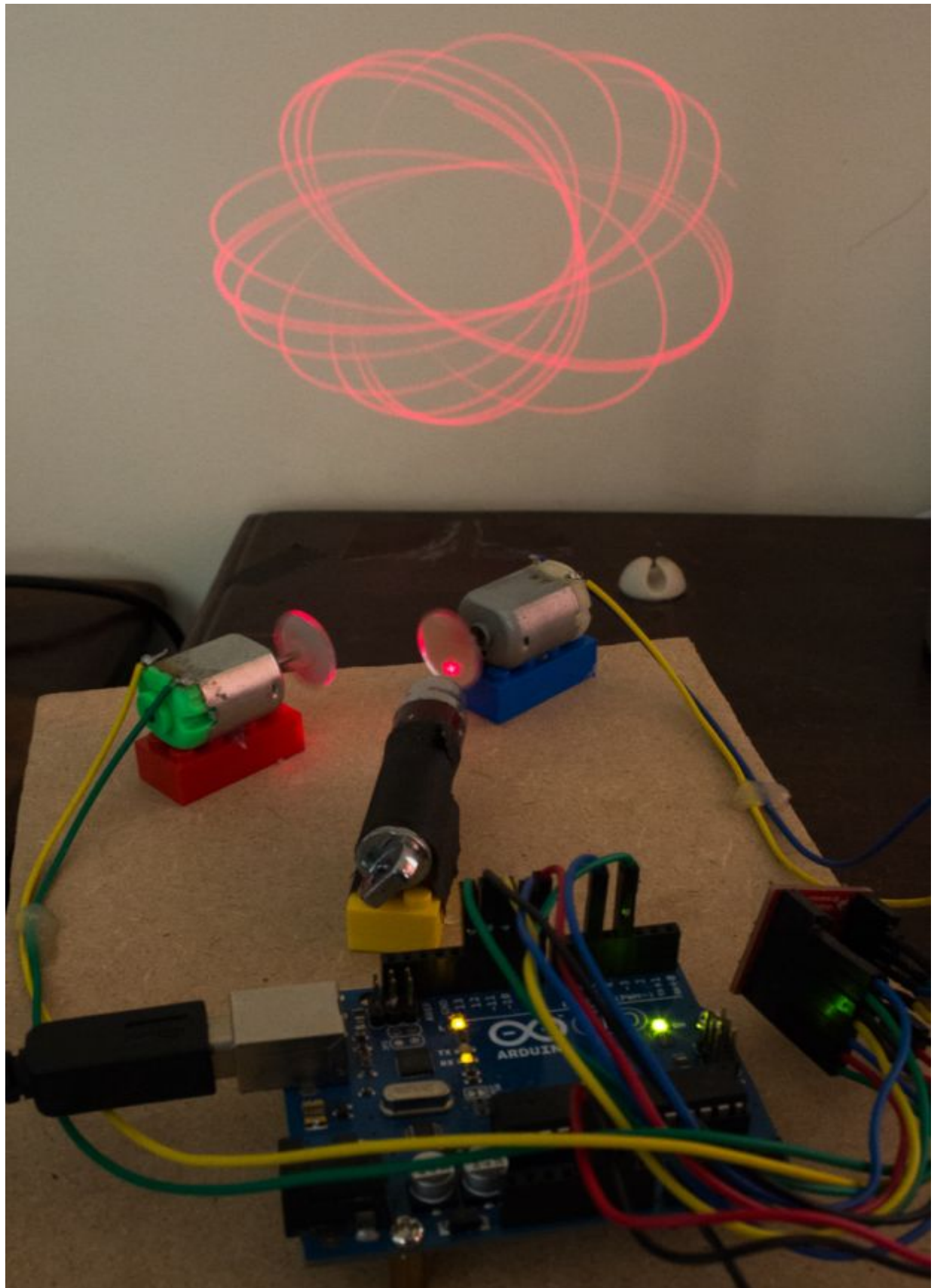
0

Input device chosen: Built-in Microph

```
opening stream...
```

13-10

1111



## 13-10 快速傅里叶变换

### 13.7 快速傅里叶变换

在本章中，我们将使用Python和Arduino来演示快速傅里叶变换（FFT）。我们将使用Python的numpy库来计算FFT，并将结果上传到Arduino。我们将使用Arduino的串口通信功能来发送FFT的结果。

### 13.8 快速傅里叶变换

在本章中，我们将使用Python和Arduino来演示快速傅里叶变换（FFT）。我们将使用Python的numpy库来计算FFT，并将结果上传到Arduino。我们将使用Arduino的串口通信功能来发送FFT的结果。

1. 首先，我们将使用Python来计算FFT。我们将使用numpy库的fft函数来计算FFT。我们将使用numpy的sqrt函数来计算平方根。我们将使用numpy的mean函数来计算平均值。我们将使用numpy的x\*\*2来计算平方。

2. 然后，我们将使用Arduino来接收FFT的结果。我们将使用Arduino的串口通信功能来接收FFT的结果。我们将使用Arduino的sqrt函数来计算平方根。我们将使用Arduino的mean函数来计算平均值。我们将使用Arduino的x\*\*2来计算平方。我们将使用Arduino的RMS函数来计算RMS值。

```
rms = numpy.sqrt(numpy.mean(x**2))
```

最后，我们将使用Arduino来发送FFT的结果。我们将使用Arduino的串口通信功能来发送FFT的结果。我们将使用Arduino的sqrt函数来计算平方根。我们将使用Arduino的mean函数来计算平均值。我们将使用Arduino的x\*\*2来计算平方。我们将使用Arduino的RMS函数来计算RMS值。

在本章中，我们将使用Python和Arduino来演示快速傅里叶变换（FFT）。我们将使用Python的numpy库来计算FFT，并将结果上传到Arduino。我们将使用Arduino的串口通信功能来发送FFT的结果。

[1] 快速傅里叶变换

ON  
Arduino  
Arduino

Python  
Arduino

---

[1] “Relays and Optoisolators,” What-When-How, <http://what-when-how.com/8051-microcontroller/relays-and-optoisolators/>



本書は、Arduino Uno R3 を中心に、USB、HDMI、Arduino、Raspberry Pi、Arduino の各ボードの接続方法、

Arduino のインストール方法、Arduino の開発環境の構築方法、Arduino の開発方法、

Arduino の応用事例、DHT11、Web、Bottle Web、Internet、IP、Bottle、DHT11、flot、LED、Web、Web の各ボードの接続方法、

Python 2.7 Raspbian Python 2.7  
shell python Python 3 shell python3

## 14.1

CPU RAM USB 35  
GPIO DHT11

### 14.1.1 DHT11

DHT11 14-1 4  
VDD+ GND— DATA 4 DATA  
Adafruit Python Adafruit\_Python\_DHT  
DHT11

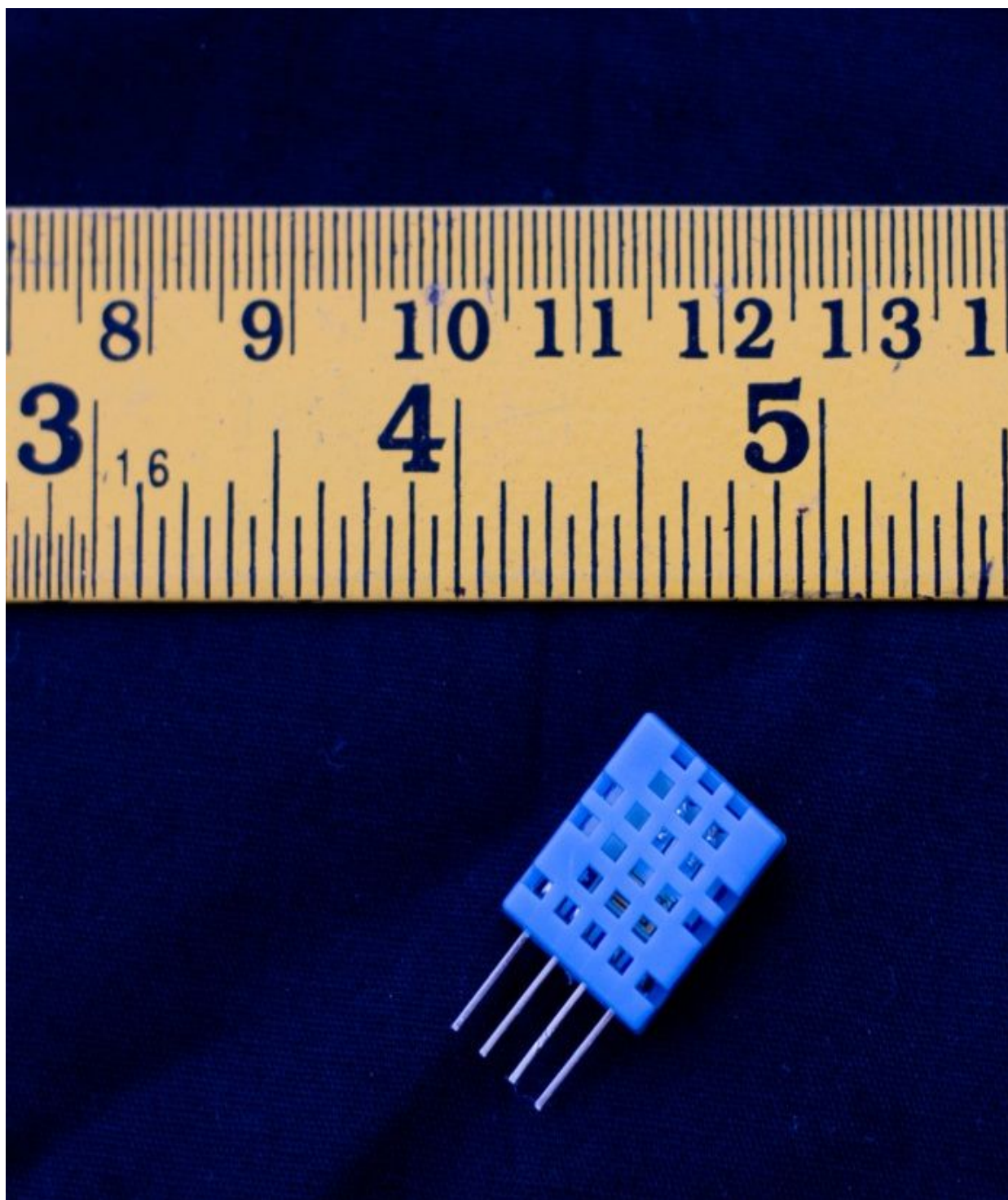


图14-1 DHT11温湿度传感器

## 14.1.2 引脚



3 Raspberry Pi 1 Model A  
+ Raspberry Pi 1 Model B + Raspberry Pi  
2 Model B Raspberry Pi Model  
B Rev 2  
Raspberry Pi Model B Rev 2  
Raspberry Pi Model B Rev 2

14-2 B USB HDMI  
USB 26 GPIO 13  
SD Broadcom  
BCM2835 ARM CPU 700  
MHz 512MB SDRAM  
Raspberry Pi Model B Rev 2  
Raspberry Pi Model B Rev 2  
Raspberry Pi Model B Rev 2

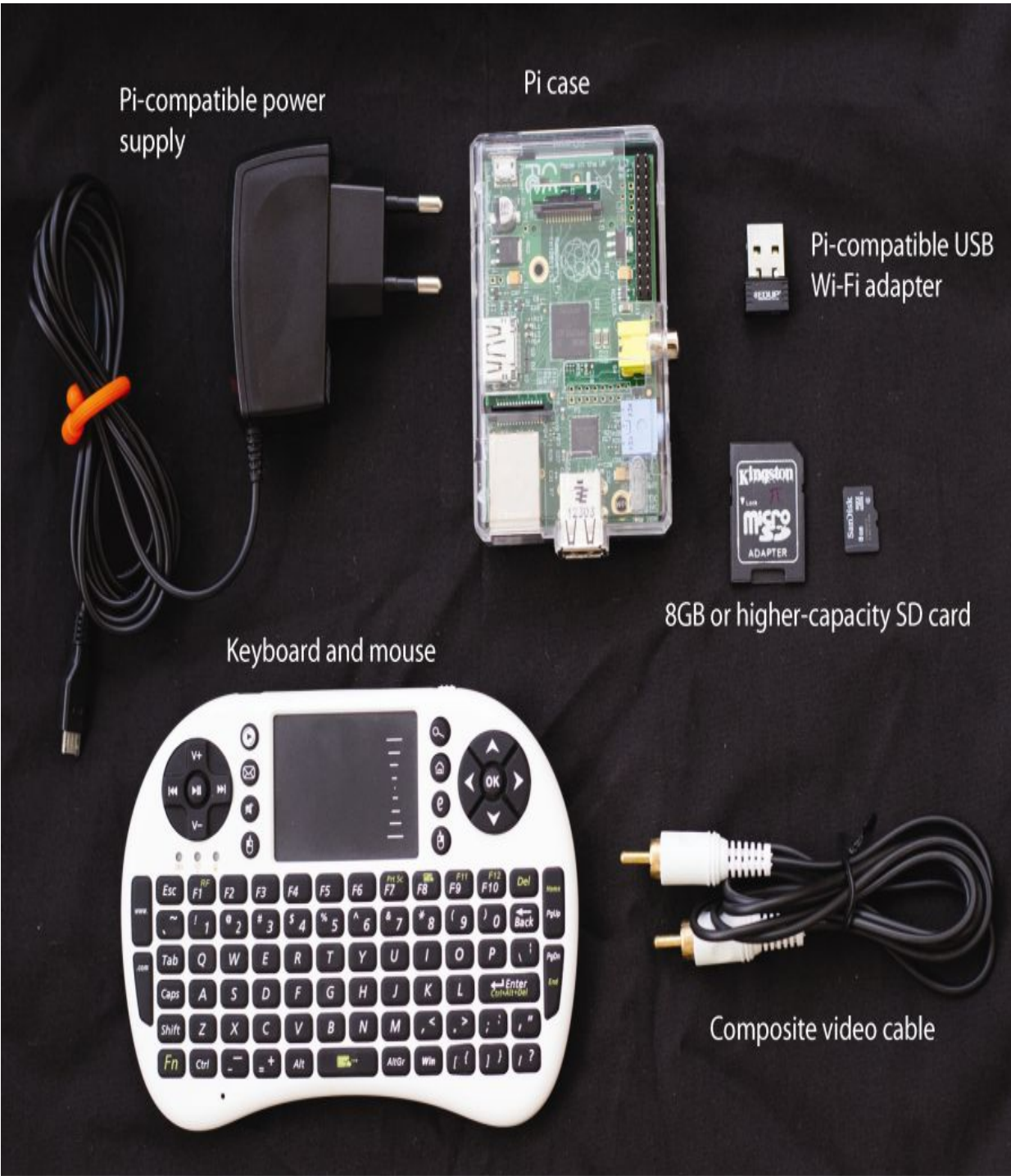


□14-2 □□□□□B

## 14.1.3 □□□□□

Arduino  
14-3  
□□□

- 8GB SD
- USB Wi-Fi
- 5V 1200 mA  
USB 5V 2500 mA
- 
- USB
- HDMI HDMI  
C



14-3



Python Pi http://elinux.org/  
RPI\_VerifiedPeripherals

## 14.2

Python  
Raspberry Pi Foundation  
“Getting Started with NOOBS”  
<http://www.raspberrypi.org/help/noobs-setup/>

### 14.2.1

SD  
Raspbian  
Linux wiki  
“RPi Easy SD Card Setup”  
[http://elinux.org/RPi\\_Easy\\_SD\\_Card\\_Setup](http://elinux.org/RPi_Easy_SD_Card_Setup)  
“Using NOOBS”

### 14.2.2

SD  
raspi-config  
Raspberry Pi Foundation  
raspi-config

<https://www.raspberrypi.org/documentation/configuration/raspi-config.md>

1 Expand Filesystem SD

2 Enable Boot to Desktop/Scratch Desktop

3 Change Time Zone  
Internationalization Options

4 Advanced Options Overscan

5 SSH Enable or Disable SSH  
Server Advanced Options

Finish

## 14.2.3 Wi-Fi

Wi-Fi  
<http://elinux.org/> Raspbian  
IP  
Nano Nano UI  
Ctrl-X Yes

How to configure network in LX Terminal on Raspbian  
How to configure network

```
$ sudo nano /etc/network/interfaces
```

How to configure interfaces in Raspbian

```
auto lo
```

```
iface lo inet loopback
```

```
iface eth0 inet dhcp
```

```
allow-hotplug wlan0
```

```
iface wlan0 inet manual
```

```
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
```

```
iface default inet static
```

```
address 192._ x.x.x _
```

```
netmask 255.255.255.0
```

```
gateway 192._ x.x.x _
```

How to configure address, netmask, gateway in  
Raspbian  
How to configure network in Linux  
Linux ifconfig  
Windows Windows R  
ipconfig/all  
OS X System Preferences-

>Network  
IP  
IP

WiFi Config  
Wi-Fi  
<https://learn.adafruit.com/> Adafruit  
Midori

## 14.2.4

RPi.GPIO  
Bottle Web  
Raspberry Pi  
Python

```
$ sudo apt-get update

$ sudo apt-get install python-setuptools

$ sudo apt-get install python-dev

$ sudo apt-get install python-rpi.gpio

$ sudo easy_install bottle
```

<http://www.flotcharts.org/> flot  
JavaScript  
flot

```
$ wget http://www.flotcharts.org/downloads/flot-x.zip

$ unzip flot-x.zip

$ mv flot myProjectDir/
```



Adafruit\_Python\_DHT  
[https://github.com/adafruit/Adafruit\\_Python\\_DHT](https://github.com/adafruit/Adafruit_Python_DHT)  
DHT11

```
$
git clone https://github.com/adafruit/Adafruit_Python_DHT.g
it
```

```
$ cd Adafruit_Python_DHT
```

```
$ sudo python setup.py install
```

## 14.2.5 SSH

Linux OS X Secure Shell  
SSH Windows PuTTY

SSH

❶ moksha:~ mahesh\$ ssh pi@192.168.4.32

❷ pi@192.168.4.32's password:

❸ pi@raspberrypi ~ \$ whoami

pi

pi@raspberrypi ~ \$

ssh pi IP  
ssh username@ip\_address  
ssh raspberry

whoami pi



C

## 14.2.6 Web Bottle

Web Bottle Python Web  
bottle.py Bottle

```
from bottle import route, run
```

```
@route('/hello')
```

```
def hello():
```

```
 return "Hello Bottle World!"
```

```
run(host='192.168.x.x', port=xxxx, debug=True)
```

Python @route URL  
run() Bottle  
IP debug True

http://192.168.4.4: 8080/hello  
Bottle "Hello Bottle World"  
Web

JavaScript XML AJAX  
Bottle AJAX  
jQuery

## Python

Python @  
"

```
@wrapper
def myFunc():
 return 'hi'
```

```
myFunc = wrapper(myFunc)
```

Python

## 14.2.7 flot

flot API  
HTML Flot  
simple-flot.html

```
<html>
```

```
<head>
```

```
 <meta http-equiv="Content-
Type" content="text/html; charset=utf-8">
```

```
 <title>SimpleFlot</title>
```

```
❶ <style>
```

```
 .demo-placeholder {
```

```
 width: 80%;
```

```
 height: 80%;
```

```
 }
```

```
</style>
```

```
❷ <script language="javascript" type="text/javascript"
```

```
 src="flot/jquery.js"></script>
```

```
<script language="javascript" type="text/javascript"
```

```
 src="flot/jquery.flot.js"></script>
```

```
<script language="javascript" type="text/javascript">
```

```
❸ $(document).ready(function() {
```

```
 // create plot
```

```
❹ var data = [];
```

```

 for(var i = 0; i < 500; i ++) {
 ❶ data.push([i, Math.exp(-
i/100)*Math.sin(Math.PI*i/10)]);
 }

 ❷ var plot = $.plot("#placeholder", [data]);

 });

</script>

</head>

<body>

 <h3>A Simple Flot Plot</h3>

 <div class="demo-container">

 ❸ <div id="placeholder" class="demo-placeholder"></div>

 </div>

</body>

</html>

```

❶ 在 CSS 中为 demo-placeholder 添加样式

❷ 在 HTML 中引入 jQuery.js 和 jQuery.flot.js 文件

❸ 在 HTML 中为 jQuery.flot 添加初始化的代码

在 JavaScript 中，我们使用 ❹ 来初始化 jQuery

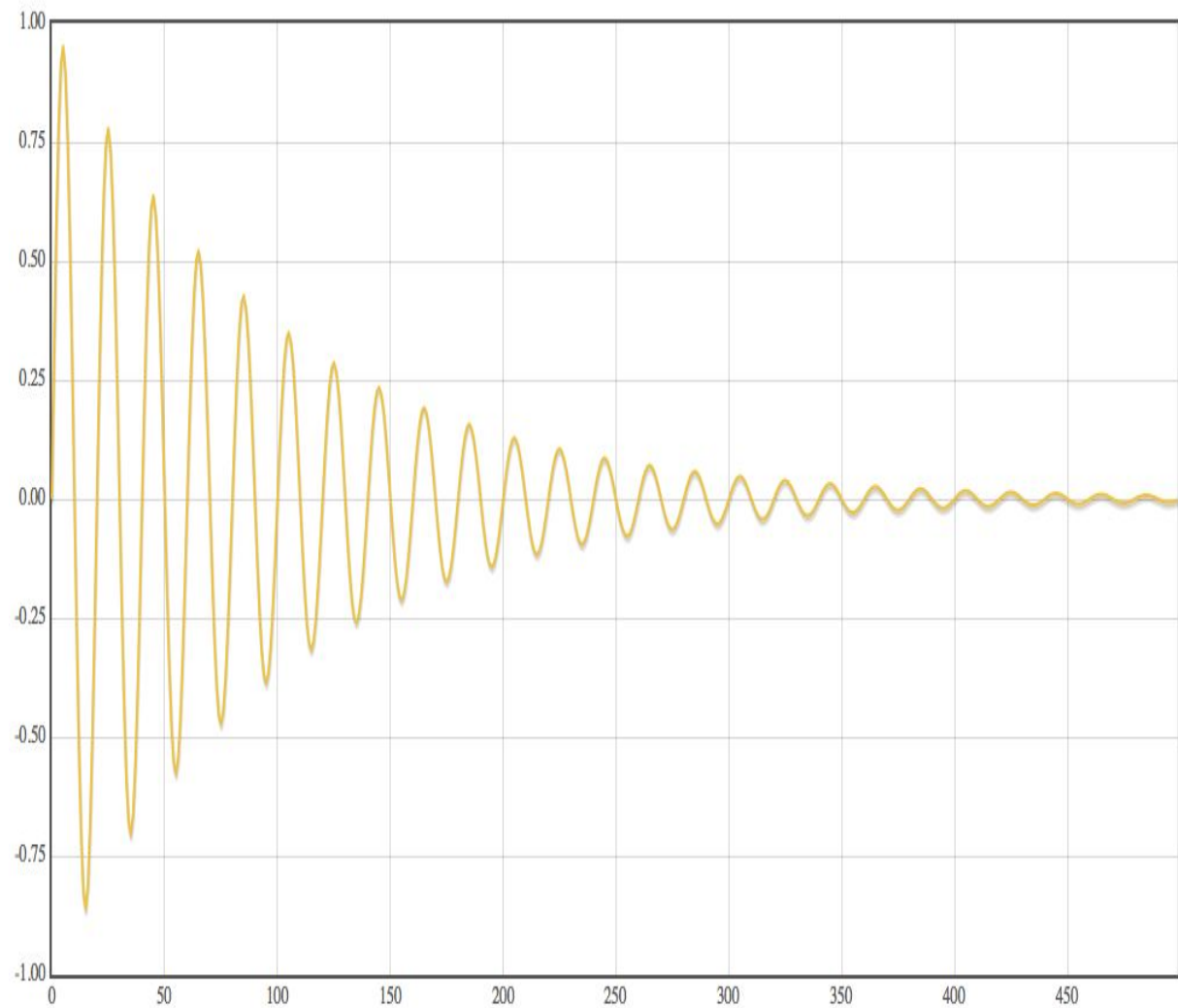
\$(document).ready() 方法会在 HTML 文档

□□□□□□□□□□□□□□□□**④**□□□□□□□□JavaScript  
□□□□□□□500□□□□□[*i*□*y*]□□□□□□□□**⑤**□□□□□□□□  
□□□□□□□□□□□□□□□□ *x* □ *y* □□□

$$x(y) = e^{\frac{-x}{100}} \sin\left(\frac{2\pi}{10}x\right) \quad x \text{ □□□□□}[0, 500]$$

□**⑥**□□□flot□□□plot()□□□□□**⑦**□□plot()□□□□□□□□  
□HTML□□□placeholder□□□□id□□□□□□□□□□□□□□□□□  
□□□HTML□□□□□□□□□□□□14-4□□□□□□□□

## A Simple Flot



### 14-4 flot

flot

flot

14.4.2

## 14.2.8

このコマンドを実行すると、システムは安全にシャットダウンします。  
SSHで接続している場合は、接続が切断されます。  
SSHで接続している場合は、接続が切断されます。

```
$ sudo shutdown -h now
```



このコマンドを実行すると、システムは安全にシャットダウンします。  
Linuxで実行します。

このshutdownコマンドを実行すると、システムは10分後に  
安全にシャットダウンします。

## 14.3 実験

この実験では、以下の部品を使用します。

- DHT11温度湿度センサー
- 4.7kΩ抵抗
- 100Ω抵抗
- LED
- 電池

14-5の回路図に従って、DHT11のVDDを+5V  
に接続し、2本のDHT11のDATAピンを  
16のDHT11のGNDピンに接続し、6本の  
DATAピンをVDDピンに4.7kΩの抵抗を  
100Ωの抵抗をGNDピンに接続し、18本の



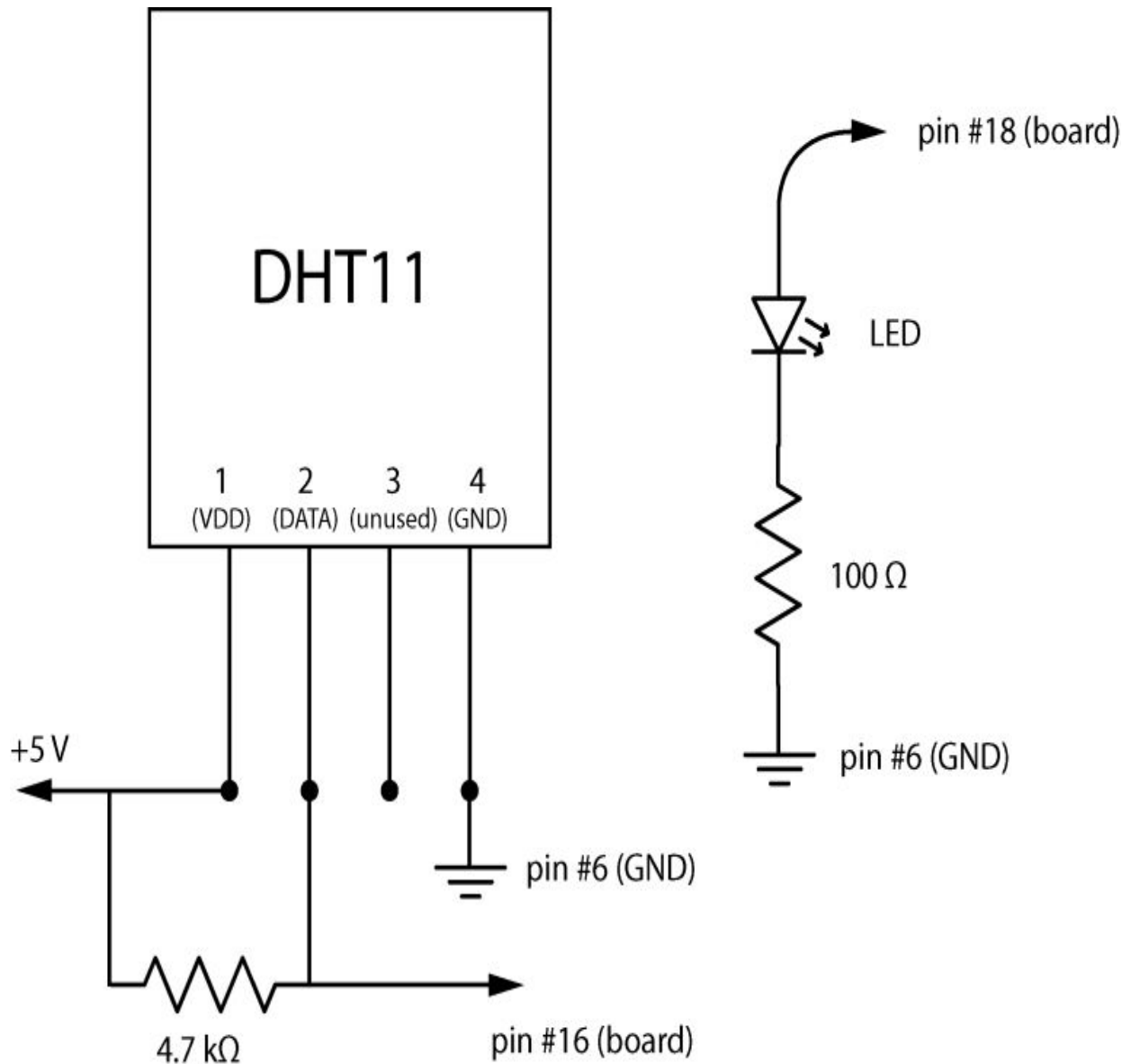
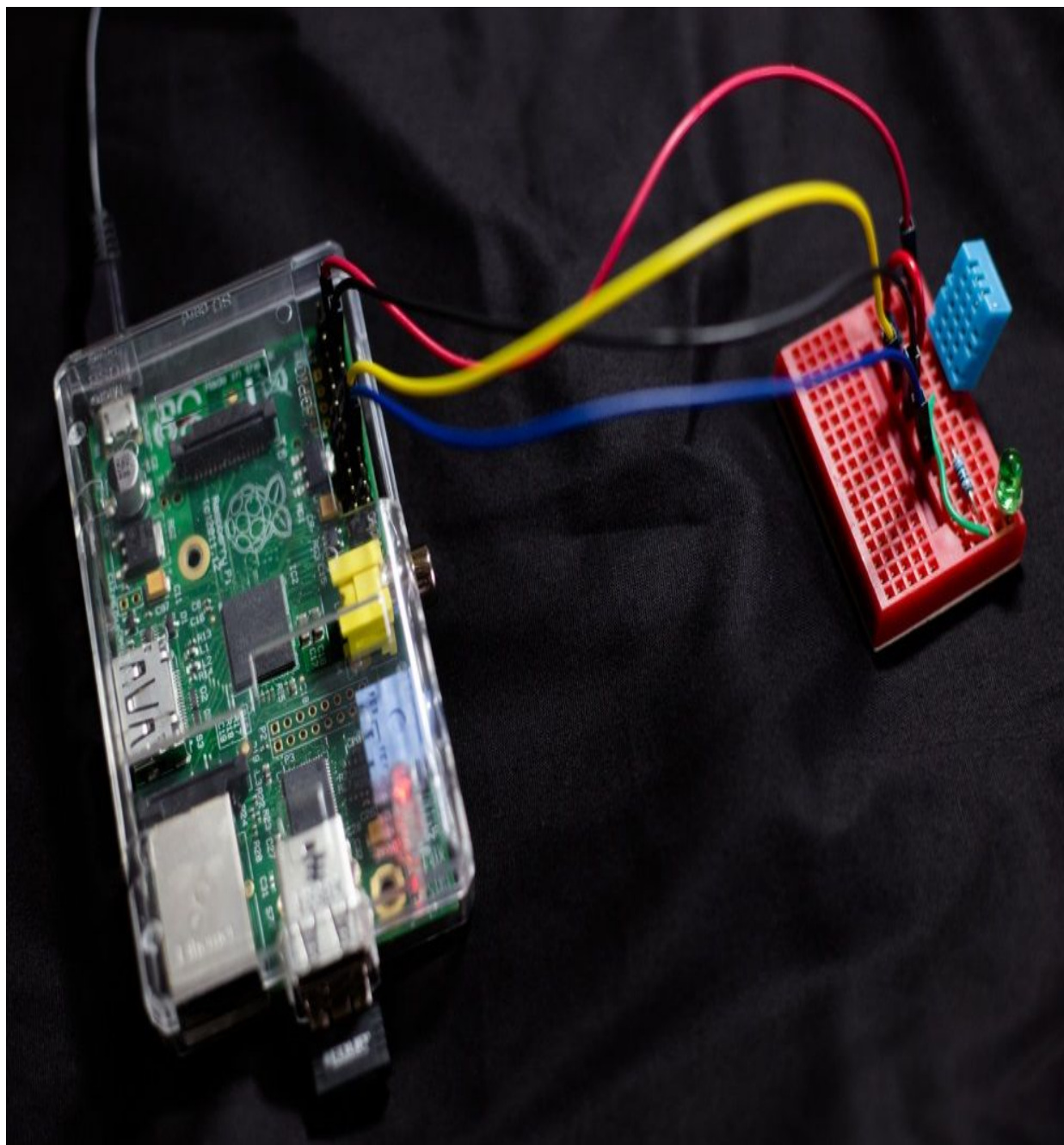


图14-5 使用DHT11和LED的电路

在本章中，我们将使用DHT11和LED来制作一个简单的温度湿度计。图14-6展示了电路的连接方式。



14-6 DHT11 LCD

## 14.4

14.5

## def main():

```
def main():
 print 'starting piweather...'

 # create parser

 ❶ parser = argparse.ArgumentParser(description="PiWeather...")

 # add expected arguments

 parser.add_argument('--
ip', dest='ipAddr', required=True)

 parser.add_argument('--
port', dest='portNum', required=True)

 # parse args

 args = parser.parse_args()

 # GPIO setup

 ❷ GPIO.setmode(GPIO.BOARD)

 ❸ GPIO.setup(18, GPIO.OUT)

 ❹ GPIO.output(18, False)

 # start server

 ❺ run(host=args.ipAddr, port=args.portNum, debug=True)
```

❶ argparse.ArgumentParser(description="PiWeather...")  
--ip IP --port port  
❷ GPIO.setmode(GPIO.BOARD)

BOARD  
③ 18 LED  
④ False LED IP  
Bottle debug True  
⑤

## 14.4.1

```
① @route('/getdata', method='GET')
② def getdata():
③ RH, T = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 23)
 # return dictionary
④ return {"RH": RH, "T": T}
```

/getdata ①  
URL /getdata getdata() ②  
Adafruit\_DHT ③ ④  
“JavaScript JSON”  
JSON —  
getdata() JSON —  
RH T

## 14.4.2

plot() HTML  
<head> “CSS”

# JavaScript

```
@route('/plot')

def plot():
 ❶ return ''

<html>

<head>

 <meta http-equiv="Content-
Type" content="text/html; charset=utf-8">

 <title>PiWeather</title>

 <style>

 .demo-placeholder {

 width: 90%;

 height: 50%;

 }

 </style>

 <script language="javascript" type="text/javascript"
src="jquery.js"></script>

 <script language="javascript" type="text/javascript"
src="jquery.flot.js"></script>

 <script language="javascript" type="text/javascript"
src="jquery.flot.time.js"></script>
```

plot() 访问 /plot URL 的 Bottle 应用 的 URL  
调用 plot() 返回 ❶ 的 plot() 返回 HTML 内容

Web HTML  
CSS flot 14-4  
flot

```
<body>HTML</body>
```

<body>

<div id="header">

## ① Temperature/Humidity

&lt;/div&gt;

<div id="content">

```
<div class="demo-container">
```

```
❷ <div id="placeholder" class="demo-
placeholder"></div>
```

&lt;/div&gt;

③ `<div id="ajax-panel"> </div>`

&lt;/div&gt;

<div>

```
4 <input type="checkbox" id="ckLED" value="on">Enable Lighti
ng.
```

5 `<span id="data-values"> </span>`

&lt;/div&gt;

&lt;/body&gt;

</html>

❶ `$(document).ready(function() {`  
❷ `var options = {`  
`<placeholder>` `$.plot(` `JavaScript` `plot`  
❸ `$("#ajax-panel", HTML)` `AJAX`  
❹ `$("#ckLED", checkbox)` `LED`  
`data-` `values` `HTML` `<script language`  
❺ `="javascript"...>` `HTML` `<head>`  
`JavaScript` `ID`

`JavaScript` `LED` `<script language`  
`=“javascript”...>` `HTML` `<head>`

❶ `$(document).ready(function() {`

`// plot options`

❷ `var options = {`

`series: {`

`lines: {show: true},`

`points: {show: true}`

`},`

❸ `grid: {clickable: true},`

❹ `yaxes: [{min: 0, max: 100}],`

`xaxes: [{min: 0, max: 100}],`

`};`

```
// create empty plot
```

```
❶ var plot = $.plot("#placeholder", [[]], options);
```

HTML❶ready()❷  
options❸  
❹  
3❺plot()❻ID  
options❼

JavaScript❶

```
// initialize data arrays
```

```
❶ var RH = [];
```

```
var T = [];
```

```
var timeStamp = [];
```

```
// get data from server
```

```
❷ function getData() {
```

```
 // AJAX callback
```

```
❸ function onDataReceived(jsonData) {
```

```
❹ timeStamp.push(Date());
```

```
 // add RH data
```

```
❺ RH.push(jsonData.RH);
```

```
 // removed oldest
```

```
❻ if (RH.length > 100) {
```



```

 RH.splice(0, 1);
 }
 // add T data
 T.push(jsonData.T);
 // removed oldest
 if (T.length > 100) {
 T.splice(0, 1);
 }
7 s1 = [];
 s2 = [];
 for (var i = 0; i < RH.length; i++) {
 s1.push([i, RH[i]]);
 s2.push([i, T[i]]);
 }
 // set to plot
8 plot.setData([s1, s2]);
 plot.draw();
 }

 // AJAX error handler
9 function onError(){
 $('#ajax-panel').html('< p> Ajax error! < /p>');
 }

```



❶ `plot()` 3

$[[[i_0, RH_0], [i_1, RH_1], \dots], [[i_0, T_0], [i_1, T_1]]]$

❷

❸ AJAX ID `ajax-panel` HTML ❹ AJAX `URL` `getdata` `Bottle` HTTP `GET()` `json` AJAX `OnDataReceived()` `onError()` AJAX

### 14.4.3 `update()`

`getData()` `update()`

```
// define an update function
function update() {
 // get data
❶ getData();
 // set timeout
❷ setTimeout(update, 1000);
}
```

```
// call update
```

```
update();
```

update() 함수는 ❶ getData() 함수를 ❷ 호출하여  
JavaScript의 setTimeout() 함수를 1000ms로 설정하여  
getData() 함수를 호출하는 코드를 작성합니다.

## 14.4.4 LED를 JavaScript로 제어

LED를 JavaScript로 제어하는 Web 페이지에  
AJAX를 사용합니다.

```
// define the click handler for the LED control button
```

```
❶ $('#ckLED').click(function() {
 ❷ var isChecked = $("#ckLED").is(":checked") ? 1:0;
 ❸ $.ajax({
 url: '/ledctrl',
 type: 'POST',
 data: { strID:'ckLED', strState:isChecked }
 });
});
```

❶ HTML의 ID가 ckLED인 LED 버튼을 클릭하면  
jQuery의 ajax() 함수를 사용하여 URL /ledctrl로 HTTP  
POST로 AJAX 요청을 보냅니다.

通过AJAX控制GPIO引脚

```
❶ @route('/ledctrl', method='POST')

 def ledctrl():

❷ val = request.forms.get('strState')

❸ on = bool(int(val))

❹ GPIO.output(18, on)
```

❶ 访问URL /ledctrl Bottle 的 ledctrl() 方法  
❷ 从 Bottle 的 request 对象中获取 strState 的值  
❸ 将 strState 的值转换为布尔值  
❹ 通过 GPIO.output(18, on) 控制 LED

## 14.4.5 交互性

使用 plot() 方法可以创建交互式 plot。通过调用 plot() 方法，可以设置 plot 的交互性。通过设置 clickable: true，可以启用交互性。

```
$("#placeholder").bind("plotclick", function (event, pos, item) {

 if (item) {

❶ plot.highlight(item.series, item.datapoint);

❷ var strData = ' [Clicked Data: ' +
```

```

 timeStamp[item.dataIndex] + ': T = ' +
T[item.dataIndex] + ', RH = ' + RH[item.dataIndex]
 + ']';

```

```

❸ $('#data-values').html(strData);
 }
 });
});

```

```

</script>
</head>

```

❶ flot highlight()
 ❷ ID data-value
 HTML flot item
 dataIndex ready()
 HTML ❸

Python JavaScript
 Bottle Web JavaScript

```

@route('/<filename:re:.*\.js>')

def javascripts(filename):

 return static_file(filename, root='flot')

```

Bottle float/

## 14.5

<https://github.com/electronut/pp/tree/master/piweather/piweather.py>

```
from bottle import route, run, request, response
from bottle import static_file
import random, argparse
import RPi.GPIO as GPIO
from time import sleep
import Adafruit_DHT
```

```
@route('/hello')
```

```
def hello():
```

```
 return "Hello Bottle World!"
```

```
@route('/<filename:re:.*\.js>')
```

```
def javascripts(filename):
```

```
 return static_file(filename, root='flot')
```

```
@route('/plot')
```

```
def plot():
 return '''

<html>

<head>

 <meta http-equiv="Content-
Type" content="text/html; charset=utf-8">

 <title>PiWeather</title>

 <style>

 .demo-placeholder {
 width: 90%;
 height: 50%;
 }

 </style>

 <script language="javascript" type="text/javascript"
 src="jquery.js"></script>

 <script language="javascript" type="text/javascript"
 src="jquery.flot.js"></script>

 <script language="javascript" type="text/javascript"
 src="jquery.flot.time.js"></script>

 <script language="javascript" type="text/javascript">

$(document).ready(function() {
```



```
// plot options
var options = {
 series: {
 lines: {show: true},
 points: {show: true}
 },
 grid: {clickable: true},
 yaxes: [{min: 0, max: 100}],
 xaxes: [{min: 0, max: 100}],
};

// create empty plot
var plot = $.plot("#placeholder", [[]], options);

// initialize data arrays
var RH = [];
var T = [];
var timeStamp = [];

// get data from server
function getData() {
 // AJAX callback
 function onDataReceived(jsonData) {
```

```
 timeStamp.push(Date());
 // add RH data
 RH.push(jsonData.RH);
 // removed oldest
 if (RH.length > 100) {
 RH.splice(0, 1);
 }
 // add T data
 T.push(jsonData.T);
 // removed oldest
 if (T.length > 100) {
 T.splice(0, 1);
 }
 s1 = [];
 s2 = [];
 for (var i = 0; i < RH.length; i++) {
 s1.push([i, RH[i]]);
 s2.push([i, T[i]]);
 }
 // set to plot
 plot.setData([s1, s2]);
 plot.draw();
}
```

```
// AJAX error handler

function onError(){

 $('#ajax-panel').html('<p>Ajax error!
 </p>');

}
```

```
// make the AJAX call

$.ajax({

 url: "getdata",

 type: "GET",

 dataType: "json",

 success: onDataReceived,

 error: onError

});

}
```

```
// define an update function

function update() {

 // get data

 getData();

 // set timeout

 setTimeout(update, 1000);

}
```

```
}
```

```
// call update
```

```
update();
```

```
// define click handler for LED control button
```

```
$('#ckLED').click(function() {
```

```
 var isChecked = $("#ckLED").is(":checked") ? 1:0;
```

```
 $.ajax({
```

```
 url: '/ledctrl',
```

```
 type: 'POST',
```

```
 data: { strID:'ckLED', strState:isChecked }
```

```
 });
```

```
});
```

```
$("#placeholder").bind("plotclick", function (event, pos, item) {
```

```
 if (item) {
```

```
 plot.highlight(item.series, item.datapoint);
```

```
 var strData = ' [Clicked Data: ' +
```

```
 timeStamp[item.dataIndex] + ': T = ' +
```

```
T[item.dataIndex] + ', RH = ' + RH[item.dataIndex]
```

```
 + ']';

 $('#data-values').html(strData);
 }
});
});

</script>

</head>

<body>

 <div id="header">

 <h2>Temperature/Humidity</h2>

 </div>

 <div id="content">

 <div class="demo-container">

 <div id="placeholder" class="demo-placeholder">
</div>

 </div>

 <div id="ajax-panel"> </div>

 </div>

 </div>
```

```
<input type="checkbox" id="ckLED" value="on">Enable Lightin
g.
```

```

```

```
</div>
```

```
</body>
```

```
</html>
```

```
...
```

```
@route('/getdata', method='GET')
```

```
def getdata():
```

```
 RH, T = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 23)
```

```
 # return dictionary
```

```
 return {"RH": RH, "T": T}
```

```
@route('/ledctrl', method='POST')
```

```
def ledctrl():
```

```
 val = request.form.get('strState')
```

```
 on = bool(int(val))
```

```
 GPIO.output(18, on)
```

```
main() function
```

```
def main():
```

```
print 'starting piweather...'

create parser

parser = argparse.ArgumentParser(description="PiWeather..."
)

add expected arguments

parser.add_argument('--
ip', dest='ipAddr', required=True)

parser.add_argument('--
port', dest='portNum', required=True)

parse args

args = parser.parse_args()

GPIO setup

GPIO.setmode(GPIO.BOARD)

GPIO.setup(18, GPIO.OUT)

GPIO.output(18, False)

start server

run(host=args.ipAddr, port=args.portNum, debug=True)

call main

if __name__ == '__main__':

 main()
```

## 14.6 物联网

使用DHT11模块和LCD模块，通过SSH连接到树莓派，  
并设置IP地址。

```
$ sudo python piweather.py --ip 192.168.x.x --port xxx
```

在浏览器中输入IP地址，访问天气信息。

<http://192.168.x.x:port/> plot

图14-7 天气信息

Temperature/Humidity

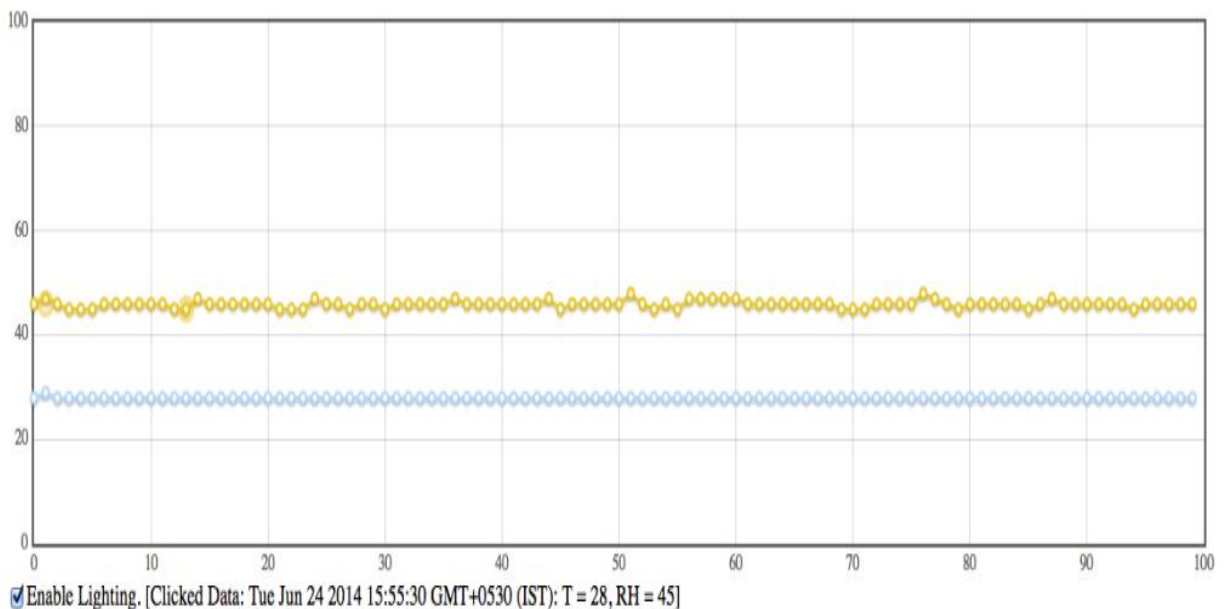


图14-7 piweather.py运行结果

在浏览器中输入IP地址，访问天气信息。  
图14-7 天气信息



Enable Lighting

## 14.7

Web

- 
- GPIO
- DHT11
- Python Web Bottle Web
- JavaScript flot
- 
- Web

## 14.8

1  
T RH /export Bottle  
CSV /plot HTML  
“Export” AJAX  
export() CSV

2 DHT11 100  
T RH

□□□□□□□□□□□□□□□□□□□□HTML□□□□□□□  
□□JavaScript□□□□□□□□□□□□□□□□100□□□  
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□  
SQLite□

## Part A 安装



本教程将指导您安装Python。本教程适用于Python 2.7.8和Python 3.3.3。本教程适用于Windows、Mac OS X和Linux。本教程适用于Python 2.7.8和Python 3.3.3。本教程适用于Python 2.7.8和Python 3.3.3。

### A.1 安装Python

请访问<https://github.com/electronut/pp/>并下载ZIP文件。

将文件解压到名为common的文件夹中。在pp-master/common文件夹中，将PYTHONPATH环境变量设置为Python的根目录。

在Windows中，将PYTHONPATH环境变量设置为Python的根目录。在Mac OS X和Linux中，将PYTHONPATH环境变量设置为Python的根目录。

`~/.profile`에 추가할 내용

```
export PYTHONPATH=$PYTHONPATH:path_to_common_folder
```

Linux에서 OS X로

```
~/.bashrc ~/.bash_profile ~/.cshrc/.login에 추가
echo $SHELL >> shell
```

Windows OS X Linux Python  
설치

## A.2 Windows

Python을 <https://www.python.org/download/>에서  
다운로드

### A.2.1 GLFW

OpenGL 3D 그래픽을 GLFW에서  
<http://www.glfw.org/download.html>에서

Windows에서 GLFW 라이브러리  
Edit Environment Variables에서  
glfw3.dll을 GLFW Python에  
C:\glfw-3.0.4.bin.WIN32\lib-  
msvc120\glfw3.dll

Python에서 GLFW를 pyglfw로  
Python에서 glfw.py를 pyglfw

common  
<https://github.com/rougier/pyglfw/>

GPU

## A.2.2

Windows Python  
32 64  
Windows

### pyaudio

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyaudio>

### pyserial

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyserial>

### scipy

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy>

<http://sourceforge.net/projects/scipy/files/scipy/>

## **numpy**

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>

<http://sourceforge.net/projects/numpy/files/NumPy/>

## **pygame**

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame>

## **Pillow**

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#pillow>

<https://pypi.python.org/pypi/Pillow/2.5.0#downloads>

## **pyopengl**

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyopengl>

## **matplotlib**

[http://www.lfd.uci.edu/~gohlke/pythonlibs/  
#matplotlib](http://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib)

matplotlib dateutil pytz pyparsing  
six

## dateutil

[http://www.lfd.uci.edu/~gohlke/pythonlibs/  
#python-dateutil](http://www.lfd.uci.edu/~gohlke/pythonlibs/#python-dateutil)

## pytz

[http://www.lfd.uci.edu/~gohlke/pythonlibs/  
#pytz](http://www.lfd.uci.edu/~gohlke/pythonlibs/#pytz)

## pyparsing

[http://www.lfd.uci.edu/~gohlke/pythonlibs/  
#pyparsing](http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyparsing)

## six

[http://www.lfd.uci.edu/~gohlke/pythonlibs/  
#six](http://www.lfd.uci.edu/~gohlke/pythonlibs/#six)

## A.2.3

Windows

<https://docs.python.org/2/install/index.html#gnu-c-cygwin-mingw>  
<http://www.scipy.org/install.html>  
Python

## A.3 OS X

OS X Python

### A.3.1 Xcode MacPorts

Xcode App Store  
Apple <https://developer.apple.com/Xcode>  
Xcode MacPorts  
MacPorts  
<http://guide.macports.org/#installing.xcode>

MacPorts Python  
OS X Python

### A.3.2

MacPorts Terminal port



如何安装Python环境

```
$ port select --list python
```

如何安装Python环境  
Python在MacPorts中安装Python 2.7

```
$ port select --set python python27
```

如何安装Python环境

```
sudo port install py27-numpy
sudo port install py27-Pillow
sudo port install py27-matplotlib
sudo port install py27-opengl
sudo port install glfw
sudo port install py27-scipy
sudo port install py27-pyaudio
sudo port install py27-serial
sudo port install py27-game
```

MacPorts中Python在/opt/local/中安装  
.profile中PATH环境变量  
Python环境变量

```
PATH=/opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin:$PATH
export PATH
```

如何安装Python包

## A.4 Linux

Linux安装Python包  
Linux安装pip包  
pip包

<http://pip.readthedocs.org/en/latest/installing.html>

pip包

```
sudo pip install matplotlib
```

matplotlib-2.0.0.tar.gz.zip包

```
sudo python setup.py install
```

matplotlib包

## B 附录

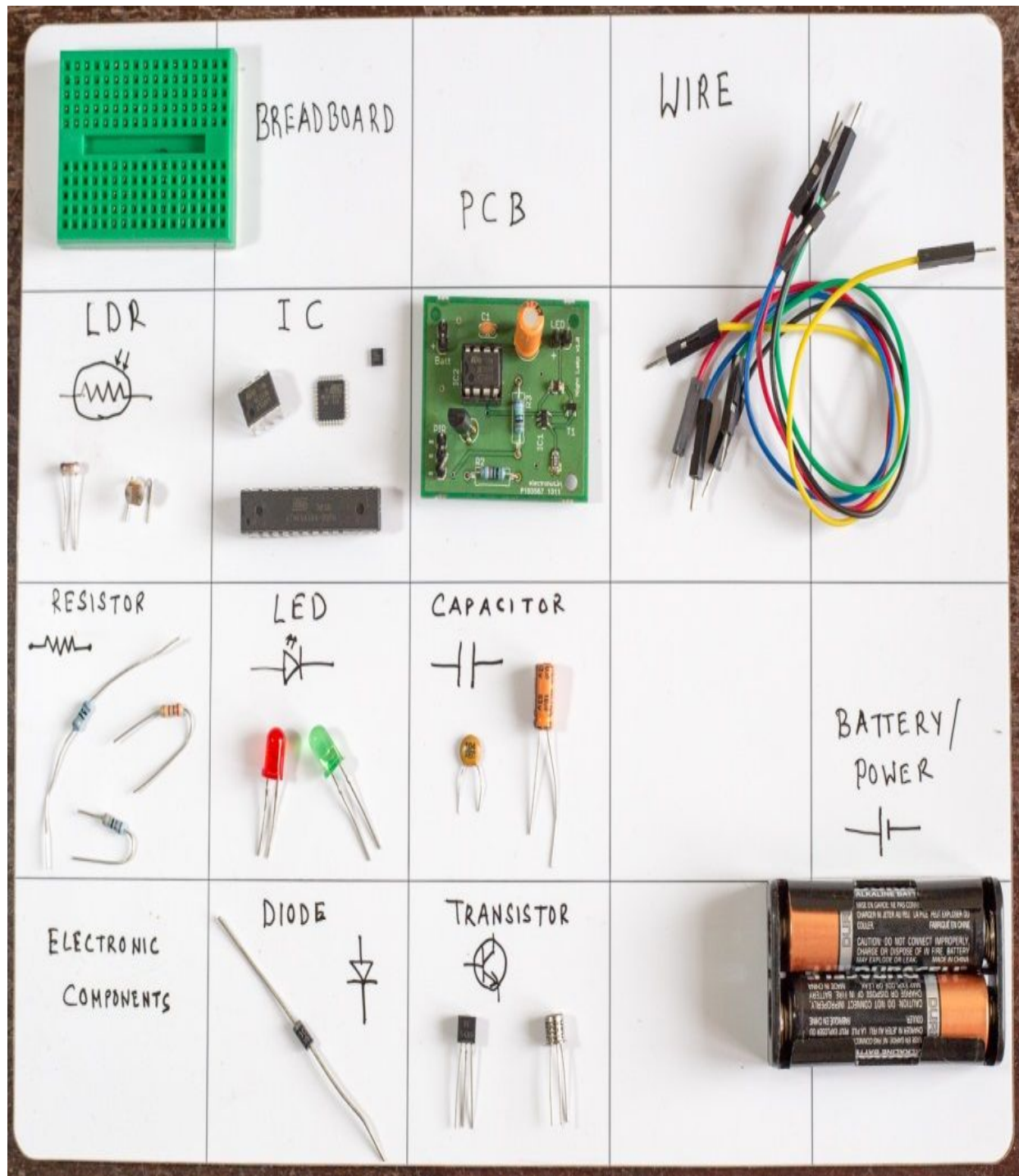
[illegible]

DIY [1] -

## B.1 ☐☐☐☐

[illegible]

**B-1**



□B-1 □□□□□□□□□□□□

**B.1.1** □□□

## B.1.2 555 LDR

### B.1.3 〇〇〇〇〇IC〇

## B.1.4 PCB

## B.1.5

[illegible]

## B.1.6 电阻

电阻器是电路中用来限制电流的元件。电阻器的阻值通常用欧姆( $\Omega$ )表示。在电路中，电阻器的符号是一个长方形，中间有一个斜线。电阻器的阻值可以在其符号旁边标注，例如  $2.7k\Omega$  表示  $2.7 \times 10^3 \Omega$ ， $2700\Omega$  表示  $2700 \Omega$ 。

## B.1.7 发光二极管(LED)

LED 是一种能够将电能转化为光能的半导体器件。LED 的符号是一个长方形，中间有一个斜线，斜线的末端有一个小圆点。LED 的符号旁边通常会标注“LED”字样。

## B.1.8 电容

电容器是电路中用来储存电荷的元件。电容器的容量通常用法拉( $F$ )表示。在电路中，电容器的符号是两个平行的长方形。电容器的容量可以在其符号旁边标注，例如  $\mu F$  表示  $10^{-6} F$ 。

## B.1.9 电感

电感器是电路中用来储存能量的元件。电感的符号是一个长方形，中间有一个斜线。电感的符号旁边通常会标注“L”字样。AC 表示交流电，DC 表示直流电。

## B.1.10 晶体管

晶体管是一种能够将电能转化为光能的半导体器件。晶体管的符号是一个长方形，中间有一个斜线。晶体管的符号旁边通常会标注“BJT”或“MOSFET”字样。

BJT MOSFET BJT MOSFET LED

### B.1.11

3 9

## B.2

B-2





PCB  
PCB  
PCB  
PCB

### B.2.3

FFT  
RMS

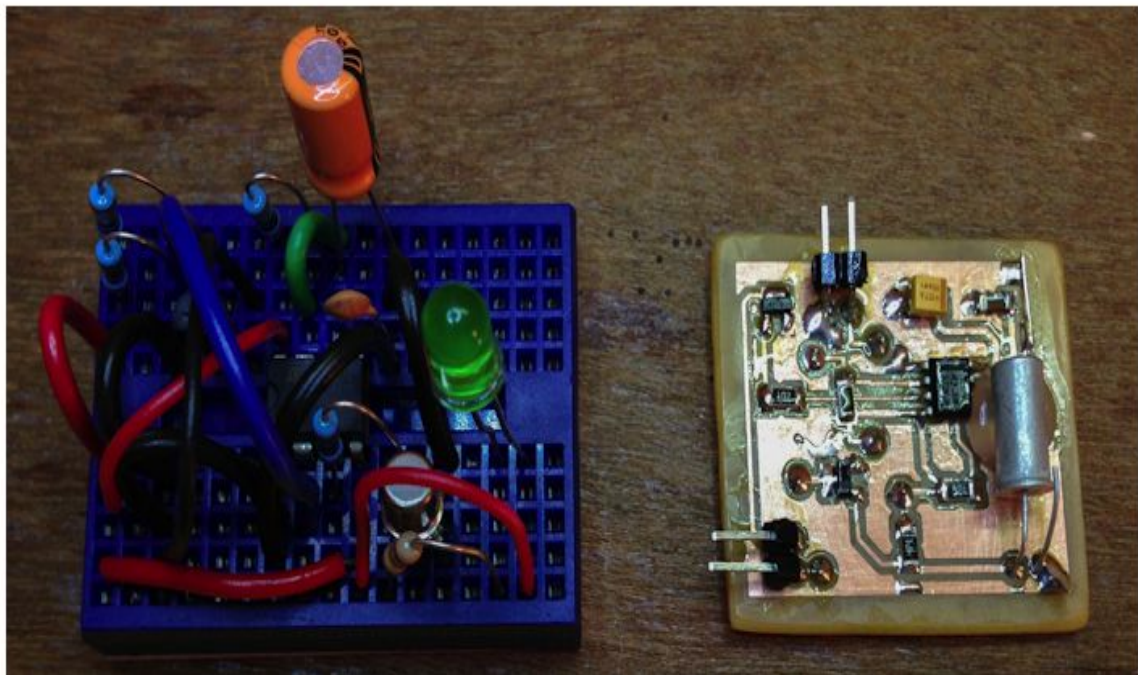
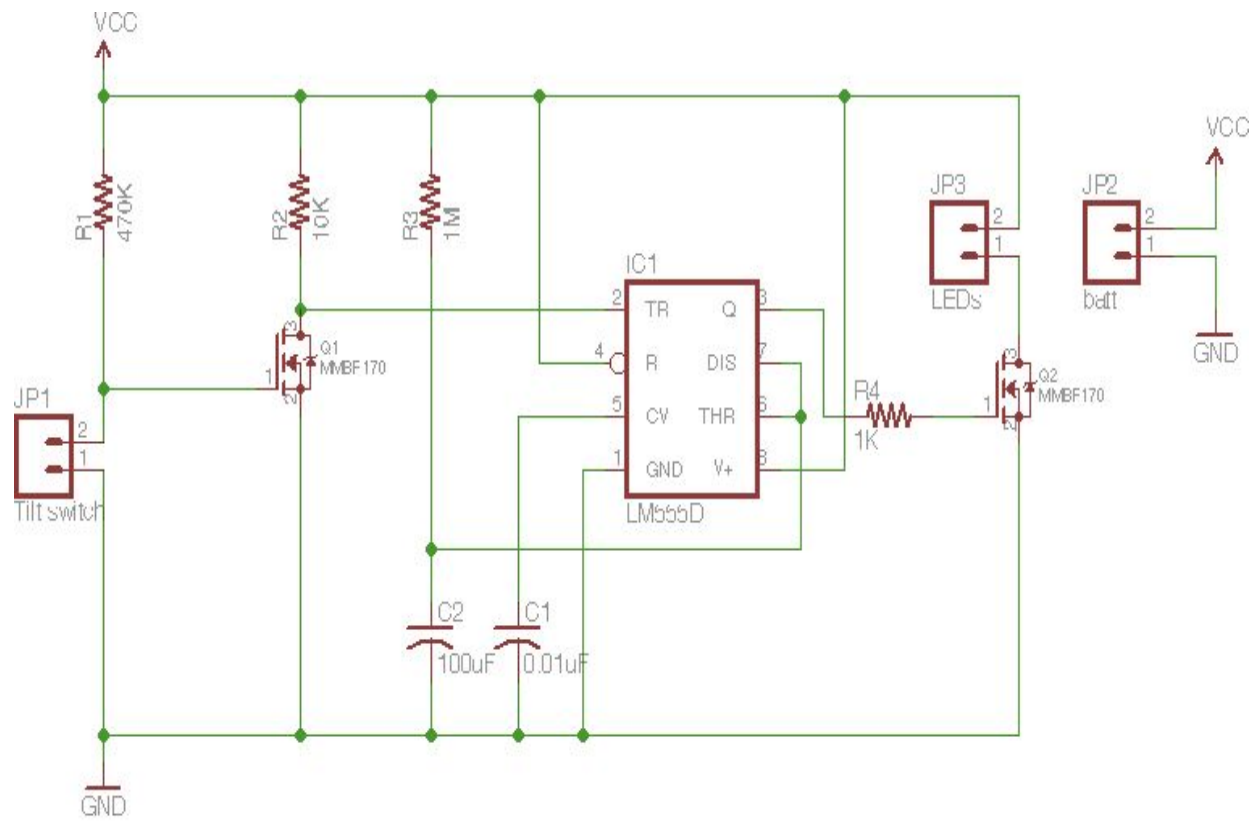
B-2  
PCB

### B.3

PCB

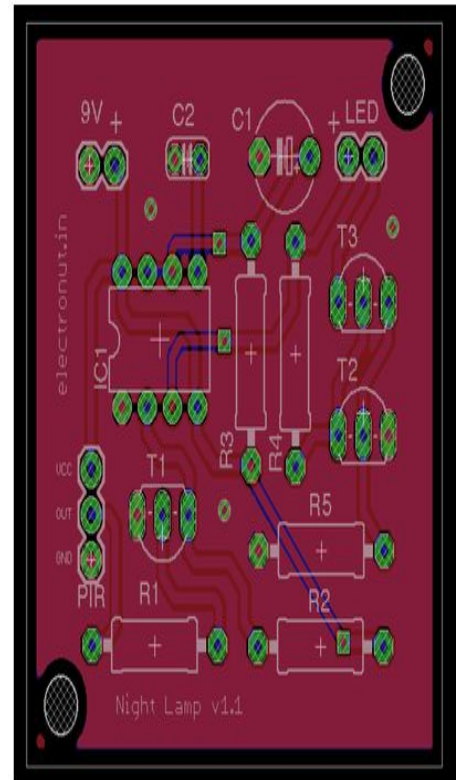
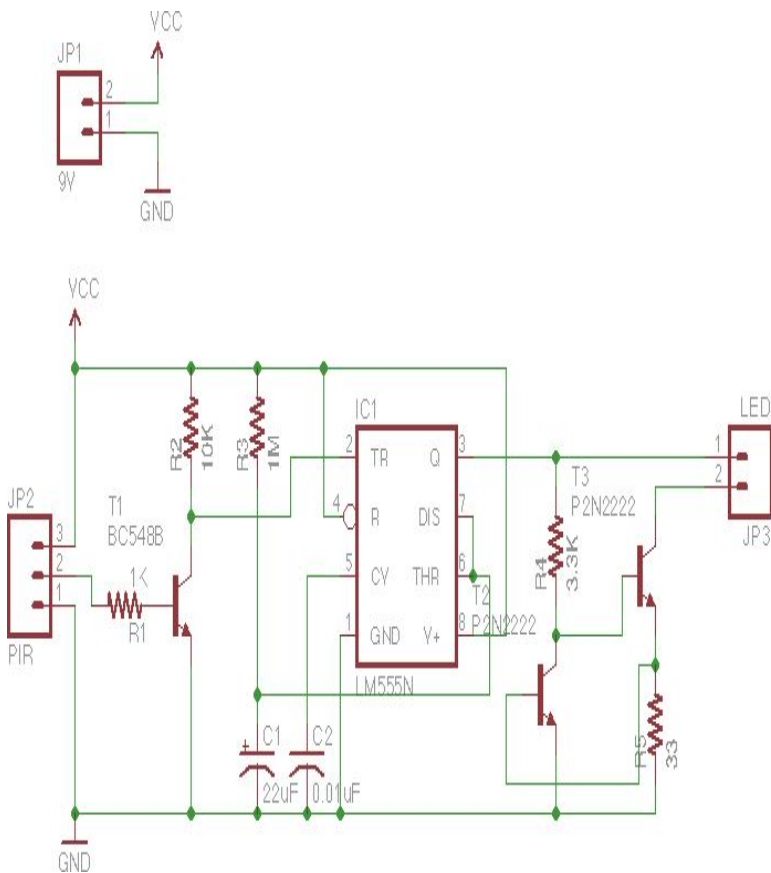
PCB  
PCB  
B-3  
PCB

PCB  
PCB EAGLE [2]  
KiCad [3] EAGLE PCB 10×8  
EAGLE [4]



图B-3 电路板PCB图

EAGLE  
EAGLE  
EAGLE  
EAGLE  
PCB  
PCB  
EAGLE  
B-4  
EAGLE  
YouTube



**B-4 EAGLE PCB**

PCB [5]  
PCB  
Gerber EAGLE

[6] PCB OSH Park  
[7]

PCB  
3D [8]  
3D [9] Rich Decibels  
[10]

## B.4

Arduino

Instructables  
<http://www.instructables.com/>  
DIY

---

[1] Paul Scherz Simon  
Monk Practical Electronics for Inventors  
3 McGraw-Hill 2013

[2] CadSoft EAGLE PCB  
<http://www.cadsoftusa.com/eagle-pcb->

[design-software/](#)

[3] KiCad EDA <http://www.kicad-pcb.org/display/KICAD/KiCad+EDA+Software+Suite/>

[4] Jeremy Blum “Tutorial 1 for Eagle Schematic Design” YouTube 2012 6 14  
<https://www.youtube.com/watch?v=1AXwjZoyNno>

[5] PCB <https://embeddedinn.wordpress.com/tutorials/home-made-single-sided-pcbs/>

[6] Akash Patel “Generating Gerber files from EAGLE” YouTube 2010 4 7  
[https://www.youtube.com/watch?v=B\\_SbQeF83XU](https://www.youtube.com/watch?v=B_SbQeF83XU)

[7] <https://oshpark.com/>

[8] 2D Inkscape <http://www.inkscape.org/en/>

[9] 3D SketchUp <http://www.sketchup.com/>

[10] Rich Decibels “Laser-Cut Project Box Tutorial” Ponoko 2011 8 9  
<http://support.ponoko.com/entries/20344437-Laser-cut-project-box-tutorial/>

## C 無線ネットワーク



14 無線ネットワーク  
無線ネットワーク  
無線ネットワーク

### C.1 Wi-Fi

14 無線ネットワーク Wi-Fi 無線ネットワーク Wi-Fi  
無線ネットワーク nano 無線ネットワーク

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
```

```
update_config=1
```

```
network={
```

```
 ssid= your-WiFi-network-name
```

```
 psk= your-password
```

```
 proto=RSN
```

```
 key_mgmt=WPA-PSK
```

```
 pairwise=TKIP
```

```
 auth_alg=OPEN
```

```
}
```

```
ssid=ssid psk=Wi-Fi
network=Wi-Fi
}
```

## C.2

```
ping
ping
```

```
$ ping 192.168.4.32
```



```
PING 192.168.4.32 (192.168.4.32): 56 data bytes

64 bytes from 192.168.4.32: icmp_seq=0 ttl=64 time=13.677 m
s

64 bytes from 192.168.4.32: icmp_seq=1 ttl=64 time=8.277 ms

64 bytes from 192.168.4.32: icmp_seq=2 ttl=64 time=9.313 ms

-- snip --
```

ping  
“Request timeout...”

## C.3 Wi-Fi

ping SSH USB Wi-Fi

```
$ sudo nano /etc/modprobe.d/8192cu.conf
```

```
disable power management

options 8192cu rtw_power_mgnt=0
```

Wi-Fi

## C.4

rsync

```
#####-n
#####"#####rsync#####
#####
#####OS X#####
```

```
#!/bin/bash
```

```
echo Backing up RPi \#1...
```

```
set this to Raspberry Pi IP address
```

```
PI_ADDR="192.168.4.31"
```

```
set this to the Raspberry Pi code directory
```

```
note that the trailing slash (/) is important with rsync
```

```
PI_DIR="code/"
```

```
set this to the local code (backup) directory
```

```
BKUP_DIR="/Users/mahesh/code/rpi1/"
```

```
run rsync
```

```
use this first to test:
```

```
rsync -uvrn pi@$PI_ADDR:$PI_DIR $BKUP_DIR
```

```
rsync -uvr pi@$PI_ADDR:$PI_DIR $BKUP_DIR
```

```
echo ...
```

```
echo done.
```

```
play sound (for OS X only)
```

```
afplay /System/Library/Sounds/Basso.aiff
```

Linux OS X `rsync`  
Windows `grsync` [\[1\]](#)

## C.5

SD  
SD  
StackExchange [\[2\]](#) Linux  
OS X `dd` Windows Win32 Disk  
Imager

## C.6

14 SSH  
SSH `ssh-keygen` /  
OS X Linux  
Windows PuTTY [\[3\]](#) IP

```
$ ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/Users/xxx/.ssh/id_rsa
):
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
```

```
Your identification has been saved in /Users/xxx/.ssh/id_rsa.
a.
```

```
Your public key has been saved in /Users/xxx/.ssh/id_rsa.pub.
b.
```

```
The key fingerprint is:
```

```
-- snip --
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
$ scp ~/.ssh/id_rsa.pub pi@192.168.4.32:~/.ssh/
```

```
The authenticity of host '192.168.4.32 (192.168.4.32)' can't
be established.
```

```
RSA key fingerprint is f1:ab:07:e7:dc:2e:f1:37:1b:6f:9b:66:
85:2a:33:a7.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added '192.168.4.32' (RSA) to the list
of known hosts.
```

```
pi@192.168.4.32's password:
```

```
id_rsa.pub 100% 398
0.4KB/s 00:00
```

```
XXXXXXXXXXXXXXXXXXXX
```

```
$ ssh pi@192.168.4.32
```

```
pi@192.168.4.32's password:
```

```
$ cd .ssh
```

\$ 1s

```
id_rsa.pub known_hosts
```

```
$ cat id_rsa.pub >> authorized_keys
```

\$ 1s

```
authorized_keys id_rsa.pub known_hosts
```

\$ **Logout**

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
This command generates an RSA key pair with a 4096-bit modulus and a comment "your_email@example.com".
The generated keys are stored in the ~/.ssh directory.
You can view the generated keys with the command:
cat ~/.ssh/id_rsa.pub
"Working with SSH Key Passphrases"
GitHub [4]
```

## C.7

```

[5]-
5
1080
30/
raspistill

```

[[6]]. [[7]]  
raspistill

## C.8

[[ALSA]] CAGE Web Design

## C.9

pyttsx  
Python

```
$
wget https://pypi.python.org/packages/source/p/pyttsx/pyttsx-1.1.tar.gz
```

```
$ gunzip pyttsx-1.1.tar.gz
```

```
$ tar -xf pyttsx-1.1.tar
```

```
$ cd pyttsx-1.1/
```

```
$ sudo python setup.py install
```

espeak

```
$ sudo apt-get install espeak
```

```
import sy:
```

```

import pyttsx

main() function
def main():
 # use sys.argv if needed
 print 'running speech-test.py...'
 engine = pyttsx.init()
 str = "I speak. Therefore. I am. "
 if len(sys.argv) > 1:
 str = sys.argv[1]
 engine.say(str)
 engine.runAndWait()

call main
if __name__ == '__main__':
 main()

```

## C.10 HDMI

HDMI
 SD
 config.txt

hdmi\_force\_hotplug=1

HDMI

## C.11 充電器

充電器は、充電ポートがmicro USBポートのAnker Astro Mini 3000mAh充電器で、充電速度は20%

## C.12 充電器

充電器は、充電ポートがmicro USBポートのAnker Astro Mini 3000mAh充電器で、充電速度は20%

```
$ cat /proc/cpuinfo
```

```
processor : 0
```

```
model name : ARMv6-compatible processor rev 7 (v6l)
```

```
BogoMIPS : 2.00
```

```
Features : swp half thumb fastmult vfp edsp java tls
```

```
CPU implementer : 0x41
```

```
CPU architecture: 7
```

```
CPU variant : 0x0
```

```
CPU part : 0xb76
```

```
CPU revision : 7
```

```
Hardware : BCM2708
```



Revision : 000f

Serial : 00000000364a6f1c

revision  
[http://elinux.org/RPi\\_HardwareHistory](http://elinux.org/RPi_HardwareHistory)  
2012 4 PCB 2.0  
B

---

[1] <http://grsync-win.sourceforge.net/>  
grsync - Windows rsync

[2] Stack Exchange SD  
“How do I backup my Raspberry Pi?”  
Stack Exchange  
<http://raspberrypi.stackexchange.com/questions/311/how-do-i-backup-my-raspberry-pi/>

[3] “How to Create SSH Keys with PuTTY to Connect to a VPS” DigitalOcean 2013 7 19  
<https://www.digitalocean.com/community/tutorials/how-to-create-ssh-keys-with-putty-to-connect-to-a-vps/>

[4] “Working with SSH Key Passphrases”  
GitHub

<https://help.github.com/articles/working-with-ssh-key-passphrases/>

[5] 如何安装和使用 Raspberry Pi 摄像头

<http://www.raspberrypi.org/product/camera-module/>

[6] The RaspberryPiGuy “Raspberry Pi-Camera Tutorial” YouTube 2013 5 26

<https://www.youtube.com/watch?v=T8T6S5eFpqE>

[7] “Raspberry Pi—Getting Audio Working”

CAGE Web Design 2013 2 9 <http://cagewebdev.com/index.php/raspberry-pi-getting-audio-working/>


[8] <https://github.com/parente/pyttsx/>  
pyttsx GitHub Python




如何安装和使用 Raspberry Pi 摄像头

如何安装和使用 Raspberry Pi 摄像头

如何安装和使用 Raspberry Pi 摄像头 (www.epubit.com.cn) IT 2015 8

如何安装和使用 Raspberry Pi 摄像头 20 IT 2015 8


异步社区  
人民邮电出版社

 书架
 (0)
 通知



[首页](#)
[图书](#)
[电子书](#)
[文章](#)
[写作](#)


# 新年新气象


社区UI全新改版，崭新面貌迎接2017！为答谢社区用户，


即日起到  
1月26号


全场电子书8折优惠！





[前端开发](#)



[数据科学](#)


[编程语言](#)



[移动开发](#)


[游戏开发](#)

[机器学习&深度学习](#)
[更多>>](#)




Python机器学习  
预测分析核心算法



贝叶斯方法  
概率编程与贝叶斯推断



机器学习项目开发实战



贝叶斯思维  
统计建模的Python学习法

免费电子书  
Free eBook

立即获取

我要写书  
Write for Us

立即查看

[Python机器学习——预测分析核心算法](#)

[贝叶斯方法：概率编程与贝叶斯推断](#)

[机器学习项目开发实战](#)

[贝叶斯思维：统计建模的Python学习法](#)

[近期活动](#)

IT Web  
1000 400

[illegible][illegible]

--	--	--	--	--	--

[illegible]

--	--	--	--	--	--	--

[illegible][illegible][illegible][illegible]

0000

“57AWG”



### Wireshark网络分析的艺术

作者：林沛满  
责编：傅道坤  
分类：计算机科学 > 安全与加密 > 网络安全

Wireshark是当前最流行的网络包分析工具。它上手简单，无需培训就可入门。很多棘手的网络问题遇到Wireshark都能迎刃而解。  
本书挑选的网络包来自真实场景，经典且接地气。讲解时采用了生活化的

[下载PDF样章](#) [配套文件下载](#)

5.6K 浏览 57 想读 7 推荐

分享：  

纸质 ¥45.00-¥31.50 (7折) 电子 ¥25.00 电子+纸质 ¥45.00

[购买](#)



+

总价：75.60

[一起购买](#)

(纸质) (纸质)

[目录](#) [评论 9](#) [勘误 1](#) [出版信息](#)

[作者简介](#) [专业书评](#) [内容提要](#)

#### 本书作者



LinPeiman  
上海  
1.0K经验值

[发私信](#) [送积分](#) [关注](#)

《Wireshark网络分析就这么简单》即《Wireshark网络分析的艺术》作者

#### 兑换样书

[立即兑换](#)

[如何赚取积分](#)

#### 电子书版本

[PDF](#) [Epub](#) [Mobi](#)

#### 精彩推荐



Nmap渗透测试指南  
作者：商广明

本书是《Python 3 入门到精通》100 个案例的  
源代码，是本书的配套资源。

□□

本书使用 Markdown 编写，所有代码均使用  
Python 3 编写，所有代码均在 Windows 10 系统下  
运行。

本书是《Python 3 入门到精通》100 个案例的  
源代码，是本书的配套资源。

□□□□□□

本书是《Python 3 入门到精通》100 个案例的  
源代码，是本书的配套资源。

□□□□

本书是《Python 3 入门到精通》100 个案例的  
源代码，是本书的配套资源。



0000



00000



00000



□□□□



QQ□□436746675

□□□□□ [www.epubit.com.cn](http://www.epubit.com.cn)>

□□□□□ □□□□



☐☐☐☐ @☐☐☐☐☐☐☐☐@☐☐☐☐☐☐☐☐-☐☐☐☐☐☐☐

☐☐&☐☐☐ contact@epubit.com.cn

## Table of Contents

[☐☐☐☐](#)

[☐☐☐☐](#)

[☐☐☐☐](#)

[☐☐](#)

[☐☐](#)

[☐☐☐☐☐☐☐☐](#)

[☐☐☐☐☐](#)

[☐☐☐☐☐☐☐☐☐☐](#)

[☐☐☐☐☐☐☐☐☐☐](#)

[☐☐☐☐☐☐☐☐☐☐](#)

[☐☐☐☐☐☐☐☐☐☐](#)

[☐☐☐☐☐☐☐☐☐☐](#)

[☐☐☐☐Python](#)

[Python☐☐☐](#)

[☐☐☐☐☐](#)

[☐☐☐☐☐☐☐☐☐](#)

## 1 iTunes

### 1.1 iTunes

### 1.2

### 1.3

#### 1.3.1

#### 1.3.2

#### 1.3.3

#### 1.3.4

#### 1.3.5

#### 1.3.6

### 1.4

### 1.5

### 1.6

### 1.7

## 2

### 2.1

#### 2.1.1

#### 2.1.2

### 2.2

### 2.3

[2.3.1 Spiro](#)

[2.3.2](#)

[2.3.3 restart\(\)](#)

[2.3.4 draw\(\)](#)

[2.3.5](#)

[2.3.6 SpiroAnimator](#)

[2.3.7 genRandomParams\(\)](#)

[2.3.8](#)

[2.3.9 update\(\)](#)

[2.3.10](#)

[2.3.11](#)

[2.3.12](#)

[2.4](#)

[2.5](#)

[2.6](#)

[2.7](#)

[3 Conway](#)

[3.1](#)

[3.2](#)

### 3.3

#### 3.3.1

#### 3.3.2

#### 3.3.3

#### 3.3.4

#### 3.3.5

#### 3.3.6

### 3.4

### 3.5

### 3.6

### 3.7

## 4 Karplus-Strong

### 4.1

#### 4.1.1

#### 4.1.2

#### 4.1.3

### 4.2

### 4.3

#### 4.3.1

#### 4.3.2

### [4.3.3 WAV](#)

### [4.3.4 pygame WAV](#)

### [4.3.5 main\(\)](#)

## [4.4](#)

## [4.5](#)

## [4.6](#)

## [4.7](#)

## [5](#)

## [5.1](#)

## [5.2](#)

## [5.3](#)

### [5.3.1](#)

### [5.3.2](#)

### [5.3.3](#)

### [5.3.4](#)

### [5.3.5](#)

### [5.3.6](#)

### [5.3.7](#)

### [5.3.8 Boids](#)

## [5.4](#)

## 5.5 関数呼び出し

## 5.6 関数

## 5.7 関数

## 関数と変数

## 第6章 ASCII文字列

## 6.1 文字列

## 6.2 文字列

## 6.3 関数

### 6.3.1 文字列の長さを取得する関数

### 6.3.2 文字列のコピー関数

### 6.3.3 文字列のASCII文字列に変換する関数

### 6.3.4 文字列の比較関数

### 6.3.5 ASCII文字列の逆変換関数

## 6.4 文字列

## 6.5 ASCII文字列の逆変換関数

## 6.6 関数

## 6.7 関数

## 第7章 文字列

## 7.1 文字列

### 7.1.1 文字列のコピー関数

[7.1.2 詳細](#)

[7.1.3 詳細](#)

[7.2 詳細](#)

[7.3 詳細](#)

[7.3.1 詳細](#)

[7.3.2 詳細](#)

[7.3.3 詳細](#)

[7.3.4 詳細](#)

[7.3.5 詳細](#)

[7.3.6 詳細](#)

[7.3.7 詳細](#)

[7.3.8 詳細](#)

[7.4 詳細](#)

[7.5 詳細](#)

[7.6 詳細](#)

[7.7 詳細](#)

[8 詳細](#)

[8.1 詳細](#)

[8.1.1 詳細](#)

[8.1.2 詳細](#)

## 8.2 背景

## 8.3 目的

### 8.3.1 目的の明確化

### 8.3.2 目的の達成方法

### 8.3.3 目的の達成手段

### 8.3.4 目的の達成手段

## 8.4 背景

## 8.5 目的の達成方法

## 8.6 目的

## 8.7 目的

## 目的の達成手段

## 9 目的の達成手段

### 9.1 目的の達成手段

### 9.2 目的の達成手段

#### 9.2.1 目的の達成手段

#### 9.2.2 目的の達成手段

#### 9.2.3 目的の達成手段

#### 9.2.4 目的の達成手段

#### 9.2.5 目的の達成手段

#### 9.2.6 目的の達成手段



### 9.3 □□□□

### 9.4 □□

#### 9.4.1 □□OpenGL□□

#### 9.4.2 □□□□

#### 9.4.3 Scene□

### 9.5 □□□□

### 9.6 □□OpenGL□□□□

### 9.7 □□

### 9.8 □□

## □10□ □□□□

### 10.1 □□□□

#### 10.1.1 □□□□□□□□

#### 10.1.2 □□□□□□

#### 10.1.3 □□□□

#### 10.1.4 □□OpenGL□□□□□□□□□□

#### 10.1.5 □□□□□

#### 10.1.6 □□□□□□

### 10.2 □□□□

### 10.3 □□□□□□□□

#### 10.3.1 □□□□□□□□□□

### 10.3.2 □□□□□□□□□□

### 10.3.3 □□□□□□□□

### 10.3.4 □□□□□□□□

### 10.3.5 □□□□□□□□

### 10.3.6 □□

### 10.3.7 Camera

## 10.4 □□□□□□□□

10.5 □□□□

## 10.6 □□□□□

### 10.6.1 □□□□□□□□

### 10.6.2 〇〇〇〇〇〇

### 10.6.3 □□□□□□□□

## 10.7 □□□□□□□□

10.8 □□□□

## 10.9 □□

10.10 □□

11000

## 11.1 □□□□

### 11.1.1 1111

### 11.1.2 数据流图

### [11.1.3 OpenGL](#)

## [11.2](#)

## [11.3](#)

## [11.4](#)

## [11.5](#)

## [11.6](#)

### [11.6.1](#)

### [11.6.2](#)

### [11.6.3](#)

### [11.6.4](#)

### [11.6.5](#)

### [11.6.6](#)

## [11.7](#)

## [11.8](#)

### [11.8.1](#)

### [11.8.2](#)

## [11.9](#)

## [11.10](#)

### [11.10.1](#)

### [11.10.2](#)

### 11.10.3 関数呼び出し

### 11.11 変数宣言

### 11.12 初期値

### 11.13 変数宣言

### 11.14 初期値

### 11.15 変数

### 11.16 変数

### 関数宣言

## 12 Arduino

### 12.1 Arduino

### 12.2 Arduino

#### 12.2.1 変数

#### 12.2.2 IDE

#### 12.2.3 変数

#### 12.2.4 変数

### 12.3 関数

### 12.4 関数

#### 12.4.1 関数

#### 12.4.2 Arduino

#### 12.4.3 関数

[12.5 Python](#)

[12.6 Python](#)

[12.7](#)

[12.8](#)

[12.9](#)

[13](#)

[13.1](#)

[13.1.1](#)

[13.1.2](#)

[13.2](#)

[13.2.1](#)

[13.2.2](#)

[13.3 Arduino](#)

[13.3.1 Arduino](#)

[13.3.2](#)

[13.3.3](#)

[13.4 Python](#)

[13.4.1](#)

[13.4.2](#)

[13.4.3 FFT](#)

[13.4.4 FFT](#)

[13.4.5](#)

[13.4.6](#)

[13.4.7](#)

[13.4.8](#)

[13.5 Python](#)

[13.6](#)

[13.7](#)

[13.8](#)

[14](#)

[14.1](#)

[14.1.1 DHT11](#)

[14.1.2](#)

[14.1.3](#)

[14.2](#)

[14.2.1](#)

[14.2.2](#)

[14.2.3 Wi-Fi](#)

[14.2.4](#)

[14.2.5 SSH](#)

### [14.2.6 Web Bottle](#)

### [14.2.7 flot](#)

### [14.2.8](#)

## [14.3](#)

## [14.4](#)

### [14.4.1](#)

### [14.4.2](#)

### [14.4.3 update\(\)](#)

### [14.4.4 LED JavaScript](#)

### [14.4.5](#)

## [14.5](#)

## [14.6](#)

## [14.7](#)

## [14.8](#)

## [A](#)

### [A.1](#)

### [A.2 Windows](#)

#### [A.2.1 GLFW](#)

#### [A.2.2](#)

#### [A.2.3](#)

## A.3 OS X

### A.3.1 Xcode MacPorts

### A.3.2

## A.4 Linux

## B

### B.1

#### B.1.1

#### B.1.2 LDR

#### B.1.3 IC

#### B.1.4 PCB

#### B.1.5

#### B.1.6

#### B.1.7 LED

#### B.1.8

#### B.1.9

#### B.1.10

#### B.1.11 /

### B.2

#### B.2.1

#### B.2.2



### B.2.3 無線LAN

#### B.3 無線LAN

#### B.4 無線LAN

### 無線LAN C 無線LAN

#### C.1 無線LAN Wi-Fi

#### C.2 無線LAN 無線LAN

#### C.3 無線LAN Wi-Fi 無線LAN

#### C.4 無線LAN 無線LAN

#### C.5 無線LAN 無線LAN

#### C.6 無線LAN SSH 無線LAN

#### C.7 無線LAN 無線LAN

#### C.8 無線LAN 無線LAN

#### C.9 無線LAN 無線LAN

#### C.10 無線LAN HDMI 無線LAN

#### C.11 無線LAN 無線LAN

#### C.12 無線LAN 無線LAN

### 無線LAN 無線LAN

#### 無線LAN 無線LAN

#### 無線LAN 無線LAN

#### 無線LAN

□□□□

□□□□□□

□□□□□□□

□□□□□□□□

□□□□□□□□□□

□□□□

□□

□□□□□□□

□□□□